THE SECRET GUIDE

TO COMPUTERS

VOLUME 1: THE MAIN SECRETS

The Secret Guide to Computers consists of two volumes. Volume 1 (The Main Secrets) covers everything that's important about computers; volume 2 (Deep Secrets) digs even deeper. You have in your hands volume 1.

The two-volume set covers the entire computer industry. It explains how to buy, fix, program, apply, and manage computers. It starts at the very beginning, but digs deep. It's easy to read. It covers practically everything you want to know; and if you have any more questions, just call the author, Russ Walter, at his home phone number (617-266-8128), and he'll answer all your questions, free! This free consulting service has saved readers many kilohours and kilobucks. Students, programmers, consultants, and heads of computer companies call Russ every day, to get his advice. You can call him 24 hours a day: he's almost always in, and he sleeps only lightly. He's been called "Boston's computer guru"; you too can call the guru, day or night.

This is the only computer book ever published that's been praised extensively by all the major computer magazines. Open this book, and you'll see why. Bring it home with you, and you'll find it a lifelong friend. Many leaders of the computer industry launched their careers by reading this book and calling Russ on the phone; now you can do the same.

In volume 1, the first chapter summarizes the entire computer industry. The second chapter explains how to program. The third chapter explains how to select equipment and compares manufacturers. The later chapters in volume 1 dig deeper, and the appendix explains the programming peculiarities of each manufacturer (IBM, Apple, Radio Shack, Commodore, CP/M computers, and oodles of others).

This book costs only $14. It contains more info bits per buck than any other computer book ever published. Get it.

Eleventh edition © 1984 by Russ Walter, $14. To order, use coupon on the back page.

Donated to
Internet Archive by
Kay Savetz

Dear You!

Congratulations! You've decided to learn about computers! That's reasonable—especially when you consider that computers all over the country are trying to learn about *you*.

This book will turn you into a computer expert, even if you're an idiot. In fact, *some of the most famous computer experts in the world have read this book*—and are idiots!

By reading this book, you'll become a computer expert, but you'll be sorry. When your friends beg you to help fix *their* computers, you'll wish you knew less.

This book tells you more about computers than your mom recommends. It explains the joys and sorrows, the highs and hassles, the philosophies and technicalities, the hardware and software, the theory and practice, the elements and excrement. At times, you'll have a ball. At other times, you'll have a ball and chain. You'll giggle and gag, laugh and lament, frolic and frown. You'll like some parts of this book so much that you'll shower me with tulips; other parts will make you shower me with tomatoes. Some people think this book's too Puritan; others think it's too pornographic.

But one thing I can assure you. You won't be bored!

This is the only book ever published that's been extensively praised by *all* the popular computer magazines. It's the only book that covers *all* aspects of computers. It's the only book that covers even the most advanced topics in a style that's easy to read. It's the only major book whose author is stupid enough to give out his home phone number.

I'm the author. My home phone number is 617-266-8128. When you have a question about computers—or about life—call me, and I'll help you, free, even if your question is weird or personal. You can call 24 hours: I'm almost always in, and I sleep only lightly. Because of the midnight phone calls, I'm not married . . . unless you count the computers I've fallen in love with.

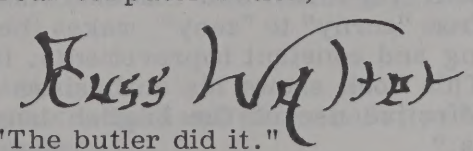
Are you a monomaniac? (A *monomaniac* is a person who thinks about just one thing. Heh, heh!)

Some monomaniacs want to know just which computer to buy. Other monomaniacs want to know just how to program. To appeal to both groups of monomaniacs, the chapters alternate between those two topics: the odd-numbered chapters explain "which to buy"; the even-numbered chapters explain "how to program". So if you have an odd personality, read just the odd-numbered chapters; and if you have an even weirder personality, read just the even-numbered chapters. If you're a well-rounded person, read all.

You have in your hands volume 1. If you're a true masochist, and want to learn even more frightening facts about the guts and consequences of computers, read volume 2. To order it, use the coupon on the back page.

Have fun with computers. But if someday you look at your heart and find it's turned into a microchip, don't bother blaming me: I've turned into a robot already!

Cheers,
At your service,
Your computer butler,



"The butler did it."

P.S. Librarians get confused by books that have weird covers. So if you're a librarian, the following dumb details will help you fill in your file cards and not get fired. . . .

Title: The Secret Guide to Computers, eleventh edition, volume 1
So-called author: Russ Walter (known to his friends as "Russy-poo")
Publisher: the same jerk as the author
Address: Russ hides in his bedroom at 92 Saint Botolph St., Boston, MA 02116
Copywrong: © 1984 by Russ Walter
Library of Congress number: if Congress knew my number, they'd censor this book, so I ain't gonna tell'em
International Standard Book Number: ha ha ha! You think this book is standard?

PRAISED BY MAGAZINES

The Secret Guide to Computers has been praised by all the famous microcomputer magazines. . . .

PERSONAL COMPUTING MAGAZINE SAYS:

"The Secret Guide to Computers is bulging with information, and you'll spend an enjoyable session with it. Russ Walter's the only author we know who invites readers to talk back. He makes it easy by publishing his home phone number and urges readers to call him whenever they encounter a problem they can't solve. His approach to text-writing sets a new style that other authors might do well to follow. The Secret Guide is one of the most readable texts on computers that's appeared. It's readable, instructive, and downright entertaining. If more college texts were written in the Russ Walter style, more college students would reach their commencement day."

POPULAR COMPUTING MAGAZINE SAYS:

"Russ Walter, author and computer instructor, is like Charlie Lau, baseball's premier batting coach: neither is a household name, but they're both highly regarded by their peers. Walter is king of the East Coast computer cognoscenti. Almost single-handedly, he's shattered the stereotype of computer teacher as absent-minded professor. He usually attends computer shows in a full-length warlock costume, as browsers partake of his shoot-from-the-hip humor. His Secret Guide is the biggest bargain in computer tutorials in our hemisphere. The Secret Guide is imbued with the same obsessive quality found in Will and Ariel Durant's history books: Walter wants to tell the whole story, from the history of the microprocessor families to frequently encountered problems in computer languages such as ALGOL. Because Russ is very committed, his Guide contains an open invitation to phone him at all hours. If CBS ever decides to replace Andy Rooney with a '60 Minutes' computer pundit, they'd need to look no further than Russ Walter. His wry Walterian observations enliven nearly every page of his book. The Secret Guide is the first collection of computer writings that one might dare call literature. Walter couldn't have packed more practical tips into the Guide if he'd used a shoehorn, yet the writing remains graceful and witty. He obviously loves computers—and loves to write about them just as much as Red Smith loved to write about baseball—yet the icing on The Secret Guide to Computers is its freewheeling zaniness."

INTERFACE AGE MAGAZINE SAYS:

"The Secret Guide to Computers claims to cover all the facets of computers simply but thoroughly, and to be foolproof, carefully tested and revised, fun, simple to read, requiring no math background, and popular among kids and adults who want straightforward talk. The book meets all those claims. How did Russ Walter do it? His examples are simple, easy to understand, and cute. He teaches the fundamentals and much more. He's anticipated the problems that frustrate the beginner and by careful instruction, repetition and reminders minimizes them. His sense of humor, which ranges from "corny" to "zany", makes the book fun. His book's gone through careful debugging and constant improvements. He's been perfecting this guide for several years. This book shows his enthusiasm and experience. It's a best buy. It makes vivid, effective use of the English language, with clear examples, charts, and programs."

INFOWORLD SAYS:

"Russ Walter is recognized and respected in many parts of the country as a knowledgeable and effective instructor. People call him at all hours of the night from as far away as Australia for advice. Talking to him is like talking to six people at once: he's an author, publisher, lecturer, teacher, consultant, and all-around 'computer butler, at your service'. He offers a unique 24-hour telephone-consulting service, free to anyone who calls. And his book tries to answer questions he's asked most often. The Secret Guide to Computers is readable, informative, and often outrageous. It includes how to program, a comparative analysis of computer systems and products, where to buy computer products, and a wealth of other information."

COMPUTER UPDATE SAYS:

"Russ Walter is Boston's computer guru, a computer wizard who believes in providing unlimited service to his customers. His Guide is cleverly graduated, outrageous, and funny. He turns computerese into plain speaking, and makes you giggle in the process. He answers telephone calls at all hours of the day and night, just as he promises in his books. He's years ahead of the pack that claims to have ways of instructing computer novices. Though Radio Shack's own manual is a transparent effort to imitate Walter's style, it suffers from oversights that don't exist in Walter's Guide. His mix of zany humor and step-by-step instruction, offered without a hint of the condescension so evident in other manuals that attempt to follow his lead, has been unique. His amusing Guide goes along with his courses, which are popular because of their unintimidating Walterian tone."

SOFTALK SAYS:

"The Guide is funny and irreverent. The material is culled from special seminars given by Russ Walter, New England's computer guru. Many leaders of the computer industry are alumni of Russ's courses. The Guide is reminiscent of an old-fashioned medicine show: it has high energy, humor, and flair. Russ pulls no punches, firing well-deserved salvos at many sacred cows. Russ has been constantly updating this book and providing phone support for many years. The Guide has long been a cult hit. Russ calls himself 'your computer butler'. This butler comes with some of the finest references in the industry, and you know how hard it is to find decent help these days."

THE WHOLE EARTH CATALOG, in its Coevolution Quarterly update, SAYS:

"Though the personal-computer subculture continues to explode, it's still kept some of the fierce honesty for which it was noted in its early years. The Secret Guide is one of the few introductory books to carry on that tradition in print, and the only introductory survey of equipment that's kept up to date. Russ Walter jokes, bitches, enthuses, condemns, and charms. He answers readers' questions for free. He's kept that up several years without burning out. The book tells the bald truth in comprehensible language. It's an interesting discipline for the author to know that if he doesn't tell it well in the book, he'll have to tell it over and over on the phone."

MANY OTHER MAGAZINES HAVE OFFERED SIMILAR PRAISES

COMPUTE MAGAZINE calls Russ Walter "an industry leader".
BYTE MAGAZINE recommends that computerists get Russ Walter's advice.
MICROCOMPUTING MAGAZINE says, "Plan ahead: get in on the secret now."
CLASSROOM COMPUTER NEWS calls Russ's courses "intensive and inexpensive."
CREATIVE COMPUTING calls the Guide "fascinating, easy to understand, an excellent book at a ridiculously low price" and "especially endorses" the Guide.
CIDER PRESS says, "The Guide is as difficult to lay aside as a first-rate mystery. It unravels the computer's mysteries effortlessly and humorously. The Guide should be given to all beginners with the purchase of their computers."
ESQUIRE recommends the Guide over all other computer books, and says it "contains lots of fact and opinion untainted by bias."

FAN MAIL

From our readers, we've received *thousands* of letters and phone calls, praising us. For example, here are the first four we received. . . .

Head of a meat factory: "I spent $100 on books on BASIC. Yours was the only one I could understand."

Head of a computer center: "Terrific! I've been looking for good books on BASIC, and yours is as thorough as the most complete but as readable as the most elementary. Yours is better than any of the others. You have them beat by a mile."

Head of a large publishing company: "I must tell you how much I enjoyed reading your computer books. This is stranger than you'd imagine, for I find the machines deplorable. But you've taken such a refreshing, comical, and interesting approach, even the likes of me can enjoy them."

Head engineer at the atomic-weapon design headquarters: "I have a big desk piled high with a lot of publications that talk a lot but don't say anything. Your book is an exception. The Secret Guide has assumed the importance here of a very critical top-secret document. It will be required reading for the whole crew."

CONTENTS OF VOLUME 1

Volume 1 tutors you about programming, equipment, and applications.

Programming. The even-numbered chapters (2, 4, and 6) explain how to program the computer, by using BASIC. Appendix 1 explains the peculiarities of computers by Apple, Commodore, IBM, Radio Shack, DEC, Texas Instruments, Timex Sinclair, and Atari and also CP/M computers.

Equipment. Chapters 1 and 3 explain computer equipment, compare manufacturers, and tell you what to buy. Chapter 1 covers the fundamentals; chapter 3 digs deeper.

Applications. Chapter 1 summarizes business applications and tells you which business software to buy. Chapter 5 blows your mind, by showing you the many weird ways the computer will change your life; you'll even learn how the computer will help you fall in love and analyze yourself. Chapter 7 shows how the computer masters art and poetry and even becomes an "artificial intellect".

As you read, you'll want to occasionally refer to appendix 2: it contains outlines, indexes, dictionaries, models of good programming, explanations of the computer's error messages, and a coupon that lists Secret Services.

Here, in detail, is what's in each chapter of volume 1. . . .

CHAPTER 1: THE COMPUTER INDUSTRY

SUBCULTURE *(drugs, my marriage, your marriage, why buy a computer): page 12*
SIZES *(maxi-mini-micro, borderline computers, which computer's best): page 16*
TOP DOGS *(favored five, IBM, DEC, Radio Shack, Apple, Commodore, CP/M computers): page 18*
EQUIPMENT *(the typical microcomputer, how to prepare your TV, how to solve problems, RAM, tape, disks, screens, types of computers, printers): page 21*
SOFTWARE *(computer languages, what's a program, operating systems, applications packages, hardware versus software, what to buy): page 28*

CHAPTER 2: PERSONAL PROGRAMMING

WARM-UP *(what's BASIC, Microsoft BASIC, what's in this book, the keyboard, get started, be bold): page 34*
CONQUER MATH *(arithmetic, question mark, E notation, range, order of operations, parentheses): page 37*
LOVELY STRINGS *(strings, strings versus numbers, long commands on small screens): page 40*
EASY PROGRAMS *(programming, multiple copies, how to print on paper): page 42*
ERRORS *(how to correct a mistake, gripes, common bloopers, rearranging your program): page 46*
TRICKY PRINTING *(semicolons, spaces, clearing the screen): page 50*
GO ANYWHERE *(GO TO, love, smelly poetry, life is an infinite loop, computer torture, dead snakes): page 53*
VARIABLES *(numeric variables, when to use variables, string variables): page 58*
INPUT *(string input, college admissions, INPUT versus PRINT, stories, bills, numeric input, conversion): page 62*
LOGIC *(colons, END, IF . . . THEN, strange programs, your personality, fancy conditions): page 68*
ADVICE *(reminders, become an expert): page 73*

CHAPTER 3: HARDWARE

CHIP TECHNOLOGY *(PC boards, chips and DIPs, bipolar versus MOS, kinds of MOS, the 4 purposes): page 76*
CPU *(what does a CPU look like, popular microprocessors, prices): page 78*
MEMORY CHIPS *(why you need memory, packaging, RAM chips, bytes in a RAM, RAM that imitates a disk, ROM, how the computer understands BASIC): page 79*
TAPE TECHNOLOGY *(how a tape holds information, kinds of tape, caring for your tapes): page 82*
DISK TECHNOLOGY *(disks are like tapes, 5-inch floppy disks, 8-inch floppy disks, write-protect notch, 3-inch floppy disks, hard disks, head motion): page 84*
BUYING A PRINTER *(keyboard, impact printers, non-impact printers, pen plotters, character quality, feeding, width, fonts, speed, prices, interfacing): page 88*
MANUFACTURERS *(Timex Sinclair, Commodore, Texas Inst., Atari, Radio Shack, Apple, Franklin, Coleco, Kaypro, Morrow, Cromemco, Epson, IBM, DEC): page 96*

CHAPTER 4: POPULAR PROGRAMMING

AIDS *(disks, tapes, debugging, editing, renumbering, pausing, apost.): page 110*
DATA . . . READ *(lists of data, debts, diets, French colors): page 116*
FOR . . . NEXT *(counting to 100, party, squares, secret meeting, midnight, favorite color, STEP): page 122*
FANCY PRINTING *(zones, TAB, calendar, LOCATE): page 128*
ELEMENTS *(grammar, a variable is a box): page 134*
ARTS *(random numbers, pixels, music): page 136*
LOOP TECHNIQUES *(incrementing, decrementing, counting, quiz, summing, checking account, betting, series, data sums): page 141*

CHAPTER 5: AMAZING APPLICATIONS

CHANGE YOUR LIFE *(magic six, games, real-time control, communication, accounting, information retrieval, education/research): page 154*
FALL IN LOVE *(history, television love, computer-dating services, the Dartmouth dater, video dating, computer dancing, computer weddings, Love Bug, Selectrocution, Interlude, Pornopoly): page 160*
ANALYZE YOURSELF *(death, brainwaves, horoscopes, psychotherapy): page 165*

CHAPTER 6: PROFESSIONAL PROGRAMMING

SUBSCRIPTS *(single subscripts, subscripted data, strange examples, problems and solutions, multiple arrays, double subscripts, multiplication table, summing a table, string arrays): page 176*
SUBROUTINES *(simple subs, Yankee Doodle, love poem, old lady, nests): page 184*
ADVANCED MATH *(exponents, contrasts, square roots, cube roots, absolute value, how to round): page 191*
STYLE *(how to invent a program, make it efficient, don't be silly, avoid round-off, test your program, document it): page 196*

CHAPTER 7: ARTS & BRAINS

CREATE ART *(gray levels, transformation, order versus disorder, envelopes, famous art, what's art, three-dimensional drawing, new style): page 212*
BE POETIC *(The Taoist Hiroshima Elevator, bubbles, kids and physics, aerospace, Me-Books, pornography, poetic images, curses): page 234*
ANALYZE WRITING *(English poetry, American history, Bible, forgery): page 246*
TRANSLATE RUSSIAN *(why Russian, early attempts, famous errors): page 248*
LOOK BACK *(dreams, the Greeks, Ramon Lull, Karel Capek, Isaac Asimov, Norbert Wiener, Alan Turing): page 250*
PLAY GAMES *(trees, checkers, maxicomputer chess, minicomputer chess, microcomputer chess, choosing a level, why a computer, Othello, backgammon, adventure, sports, space war): page 254*
HANDLE ENGLISH *(famous attempts, baseball, exceptions, what's a story problem, arithmetic & algebra, encyclopedia, the military, common concepts, psychotherapy & geometry & German, calculus, probability): page 262*
REPLACE PEOPLE *(bartenders, doctors, tutors, robots): page 270*
CREATE INTELLECT *(categories, mental games, natural languages, interpersonal relationships, the arts, protocol methods, abstract math, GPS, vision, robots, today's research, reflexive control, misinformation, the brain): page 279*

APPENDIX 1: PECULIARITIES

APPLE*: page 288.* COMMODORE*: page 297.* IBM*: page 306.* RADIO SHACK*: page 317.* CP/M COMPUTERS*: page 328.* OFF-BEAT COMPUTERS *(DEC, Texas Instruments, Timex Sinclair, Atari): page 332.*

APPENDIX 2: LISTS

ESSENTIALS *(manufacturers, hardware, operating systems, languages, applications, Modern Microsoft BASIC, Commodore extras, IBM extras, Apple extras, Radio Shack extras, CP/M extras): page 340.* MODELS*: page 342.* ERROR MESSAGES*: page 344.* PROGRAM JARGON *(elements, loops & ends, graphics, errors, advanced programming techniques): page 346.* MEMORY JARGON *(disks, tapes, memory chips, general terms): page 348.* COUPON*: page 350.*

CONTENTS OF VOLUME 2

Volume 2 digs deeper into programming, equipment, and applications. It explains advanced programming in BASIC, FORTRAN, PASCAL, COBOL, 20 other high-level languages, and several assembly languages. It gives further details and advice about the various manufacturers and their software for word processing, accounting, and data management. It also analyzes educational software and computer-literacy curricula. It explains how to launch your own computer career, and lots more! To order, use the coupon on the back page.

OLD EDITIONS

I wrote the original edition of The Secret Guide to Computers back in 1972. It was called "edition 0" (because computer folks always count from 0 instead of from 1). Then came improvements, called "edition 1", "edition 2", etc.

Today, you're reading edition 11, which is much better than its predecessors. It consists of two fat volumes: you have in your hands volume 1.

I have a few copies of older editions left, which you can still buy. Since they're thinner than the eleventh edition, they cost less. Schools who can't afford the eleventh edition buy the tenth or ninth edition instead.

Volume 1 of the tenth edition is thin (96 pages) and lists for $3.70.

Volume 1 of the ninth edition is even thinner (64 pages) and lists for just $2.75. Its super-low price and lack of sexual language make it very popular in public schools. Moreover, we give you huge discounts: you get a 20% discount if you buy 2 copies, 30% if you buy 3, 40% if you buy 5, 50% if you buy 10, 60% if you buy 30, and 70% if you buy 120. So if you buy 120 copies, your price per copy is just 82½¢!

To buy those old editions, phone me first, to make sure they're still in stock and to find out how they differ from the newer editions.

LICENSED EDITIONS

Bates College, Scelbi Publications, Brady Publishing, and Coleco (which makes the Adam computer) all received licenses from me, to reprint tiny parts of obsolete editions of The Secret Guide to Computers. In fact, the manual that comes with the Adam computer is a revision by Coleco of a tiny part of the tenth edition; I am *not* responsible for Coleco's revision, which contains several errors and omissions.

The only organization that's received a license to reprint newer editions is Birkhauser Boston Publishing. Moreover, Birkhauser's license is an exclusive: no other organization can receive such a license without Birkhauser's permission. Birkhauser sells its volumes for $14.95 each; you'll pay 95¢ less if you buy directly from me.

FREE REPRINTS

You may reprint up to 10 pages of The Secret Guide to Computers free, and make as many copies of them as you like.

If you're a teacher, give them to your students. If you're a magazine publisher, insert them in your magazine.

Your reprint must include the following notice:

Most of this material comes from the eleventh edition of The Secret Guide to Computers, copyright 1983 by Russ Walter and reprinted with permission. To get free literature about the complete Secret Guide to Computers, phone Russ at 617-266-8128, 24 hours (he's almost always in); or send a postcard to him at 92 Saint Botolph St., Boston, MA 02116.

Phone me and say which pages you're reprinting. Then send me two copies of the reprint.

FORTHCOMING BOOKS

I'm developing the next major edition (the twelfth) and also The Secret Guide to Tricky Living. Irene Vassos is creating a children's edition, which contains less sexual language. For details, use the coupon on the back page.

STAFF

The Secret Guide to Computers was written by Russ Walter, with considerable advice from Priscilla Grogan and other members of the staff. The artwork was done by Priscilla Grogan and Russ Walter, with help from FORMATT.

My editor told me to put this page in.
You don't have to read it.

DEDICATION

I dedicate this book to the computer, without whom I'd be unemployed.

WHAT THIS BOOK WILL DO FOR YOU

This book will make you even richer than the author! Unfortunately, he's broke.

PREREQUISITE

This book was written for idiots. To see whether you can get through the math, take this test: try to count to ten, but (here's the catch!) without looking at your fingers. To remove the temptation, cut them off.

ACKNOWLEDGMENT

I'd like to thank:

my many friends, whose names I've gladly forgotten;
my students, who naturally aren't my friends;
my typewriter, which has a mind of its own;
all others who helped make this book impossible.

APOLOGY

Any original ideas in this book are errors.

DISCLAIMER

The author denies any knowledge of the scintillating illegal activities he depicts.

COPYWRONG

Reproducing this material in whole or even in itsy-bitsies, without written protection from the author's hex sign, is a no-no, punishable by death, or at least by salt in your orange juice on April First. If you're a reviewer, you may quote this wonderful book, but not if you're nasty.

FORWARD

. . . because it's too late to turn back.

ABOUT THE SO-CALLED AUTHOR

The author ("Russy-poo" Walter) and the modern computer ("stored-program computer") were both conceived in 1946. Nine months later, Russy-poo was hatched; the modern computer took a few years longer. So Russ got a head start. But the computer quickly caught up. Ever since, they've been racing against each other, to see who's smartest. The outcome's still undecided.

The race is close, because Russ and the computer have so much in common. Impartial observers say the computer "acts human" and say Russ's personality is "as dead as a computer".

Yes, Russ resembles a computer in many ways. For example, both are Jewish. The father of the modern computer was John von Neumann, a Jew of German descent who, after living in Hungary, fled the Nazis and became a famous U.S. mathematician. The father of Russy-poo Walter was Henry Walter, a German Jew who fled the Nazis and became a famous U.S. dental salesman: to dentists, he sold teeth, dental chairs, and balloons (for the kids).

To try to get ahead of the computer, Russ got his bachelor's degree in math from Dartmouth in yummy '69 and has sadly remained a bachelor ever since (unless you count the computer he got married to). After Dartmouth, he got an M.A.T. in mathematical education from Harvard. Since he went to Harvard, you know he's a genius. Like most genii, he achieved the high honor of being a junior-high teacher.

After his classes showered him with the Paper Airplane Award, he moved on, to teach at an exclusive private school, for girls who were *very* exclusive. ("Exclusive" means everyone can come except you.) That was in the Big Apple (or whatever you call it).

After teaching every grade from 2 through 12 (he taught the 2nd-grade girls how to run the computer, and the 12th graders less intellectual things), he fled reality by joining Wesleyan University's math Ph.D. program in Connecticut's Middletown (the middle of Nowhere), where, after a year-and-a-half of highbrow hoopla, he was seduced by a computer, to whom he's now happily married.

The computer he married is his competitor's daughter. Actually, they're not legally married—she's under age—but she spends most of her life near (or in) his bed; he just "shacks up" with her, and when he's away on a trip he's sometimes tempted to chat with her via radio, which is how she got the name "Radio Shack".

After the wedding, he moved with his electrifying "wife" to Northeastern University in Boston (home of the bean and the cod), where he did a hilarious job of teaching in what was lewdly known as the Department of "Graphic Science". After quitting Northeastern and also editorship of *Personal Computing*, the magazine that imitates *Playboy* but replaces the girls by computers, and which goes to 50,000 homes that have computers instead of cats, he spends his time now happily losing money by publishing this book.

The young couple returns to Wesleyan in the summers, to give their ever popular (and ever ridiculous) course on *How to Become a Computer Expert in Seven Weeks (and Regret It)*. Russ's young wife does a strip tease in front of the class, to bring the class to a climax. After the students enjoy a close-up look at her most intimate parts, she gives the students hands-on experience, to teach them a thing or two about computer anatomy.

In addition to his wife, Russ has 30 groupies, with names such as "Anita Atari", "Aphrodite the Apple", "Baby Blue Burping Bonnie", "Coco the Incredible Clown", "Jack the Shack", "Kooky Casio", "Slick Vic", and "Terrible Tina with her Texas Instruments". He shoves them into a van and they drive around the country having orgies with anyone who wishes. When Russ's nuthouse comes to your town, he'll tell your mayor that the van is just for "educational purposes"; but what an education!

ABOUT THE COMPANY

What company? C'mon over, bring a bottle of booze, and then we'll have some helluva company!

We have several offices. Come visit our Northern Office, where our cold accountant computes how much money we lose. Or visit our Home Office, which is in Russ's home: it includes our production department, which is near, or in, Russ's bed. Yes, Russ published this book himself; nobody else would dare.

We're in Boston, city of lobster-lovers. Before Russ married his computer, he dated a lobster. When you try to get Russ to talk about that lobster, he clams up, and just mutters that he doesn't like aquatic females who are always trying to claw him.

But he still has a crush on lobsters. After he writes *The Secret Guide to Computers*, he'll write a book about lobsters, because they're the ultimate challenge to man: making love to a lobster is hard; making love to a computer is automatic.

Our lab used to contain lots of lobsters, until Russ replaced them by computers. Russ calls it his "sex" lab, because he loves to get a computer turned on, by tickling the keys and fingering the input. Come to our computer peep-show, where you peep at Russ as he fondles his computers.

We offer books, courses, and free consulting. Our books are ridiculous, but so are our readers. Our courses cover everything you wish to know, plus everything you'll wish you didn't. As for our consulting, if you're stupid enough to listen to Russ, he'll be stupid enough to mouth off.

As an economy measure, Russ fired his entire staff. He replaced the staff by computers. So pardon us if we talk funny: we're robots. *(Russ thinks we're just "word processors", but we know better.)*

CHAPTER 1

THE COMPUTER INDUSTRY

SUBCULTURE

DRUGS

Computers are like drugs: you begin by spending just a little money on them, but then you get so excited by the experience—and so hooked—that you wind up spending more and more money, to feed your habit.

Your first computer experience seems innocent enough: you spend just a little money, for a cute little computer that attaches to your color television. You turn the computer on, and suddenly your television shows dazzling superhuman colors that swirl hypnotically in front of you. You say "Wow, look at all those colors!" and you feel a supernatural high. But after two months of freaking out with your new computer, the high wears off, and you ask yourself, "What can I buy that's new, exciting, and gives me an even bigger high?" So you buy more stuff to attach to your computer. Now you're in really deep, financially and spiritually; you're hooked; you've become addicted to computers. And each month you'll return to your favorite computer store, to search for an ever larger high—and spend ever more money.

Look at me, for example: I'm a typical computer junkie. I've already bought 30 computers, and I'm still going. Somebody help me! My computers have already taken over my bedroom, and I've had to create a *second* bedroom, which the computers have already taken over also! Whenever I try to go to sleep, I see those computers staring at me, their lights winking, tempting me to spend a few more hours in naughty fun, even if the sun's already beginning to rise.

If you're into drugs, you'll easily master computer jargon, because the lingo of the today's computerized techno-streets is the same as the lingo of the druggie's hypodermic needle. For example, to buy a computer, you go to a *dealer*; and when you finally start using your computer, you're called a *user*.

As you get in deeper, searching for greater and greater highs, you squander more and more of your money on computer equipment, which is called *hardware*. You stay up late at night (playing computer games, or getting the errors out of your program), so that when you come into work the next morning you're all bleary-eyed. Eventually your boss suspects your computer habit, realizes that you're not giving full attention to your current job, and fires you.

Since you have no job, and your computer bills are getting higher and higher, eventually you run out of money to spend on computers. And yet you still have that urge to continue spending more. To support your habit, you write programs and try to sell them to your friends. That makes you a "pusher". You turn your friends into addicts too, and you all join the ever-increasing subculture of "computer freaks".

The only difference between computers and drugs is that if you're into drugs people call you a "washout", whereas if you're into computers people say you have a "wonderful career".

As a computer "pusher", you can make lots of money. But to be a successful pusher, you must do it "in style". For example, you don't call yourself a "pusher"; instead, you call yourself a *computer consultant*. Yes, a computer consultant is a person who gives computer advice to other people—and pushes them into buying more computers!

If a computer consultant is kind-hearted, he'll teach the neighborhood kids about computers, free! But if he seems to be a "nice guy", don't be fooled: he's just a wolf in sheep's clothing. There's even a song about it! The song—called "The Old Dope Peddler" and written by Tom Lehrer—includes these lines:

He gives the kids free samples,
Because he knows full well
That today's young innocent faces
Will be tomorrow's clientele.

Now maybe you can understand why Apple Computer Company is trying to make Congress pass a bill that will let Apple give free samples to every school in the country—and get a tax write-off, to boot!

MY MARRIAGE

I'm married to a computer. Marrying a computer is much groovier than marrying a person. For one thing, computers are good at "getting it on": they make you feel all electric and tingly. And computers never argue: they're always ready to "do it" (except when they "have a headache").

I wanted to call this book "The *Sexual* Guide to Computers", and put a picture of me and my wife on the cover; but some parts of the country still prohibit mixed marriages. That cover would be banned in Boston, which (alas!) is where I live. So I had to play it cool, and say "Secret" Guide to Computers. But here's the real secret: this book is about sex.

YOUR MARRIAGE

The computer will fascinate you. It will seduce you into spending more and more time with it. You'll fall in love with it.

You'll even start buying it presents. You'll buy it exotic foods: you'll feed it the most expensive programs, for it to munch on. You'll buy it new clothes: you'll dress it in a pretty little cover, to keep the dust off. You'll adorn it with expensive jewels: a printer and an extra disk drive.

The computer will start demanding things from you. Just when you're in the middle of an exciting orgy, and you think the computer's the best thing that ever happened to your life, all of a sudden the computer will demand that you buy it more memory. It will refuse to let you continue your programming orgy unless you agree to its demand. And you eagerly agree.

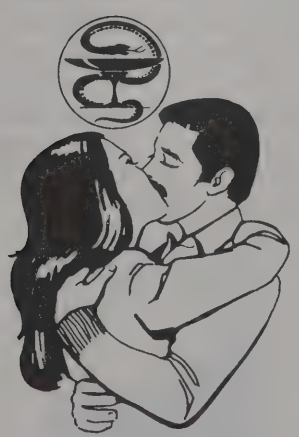
If you're a man, you'll find that the computer is a harsh mistress. If you're a woman, you'll find that the computer is an unpleasant pimp. It's always telling you to give it more, give it more. . . .

You'll feel married to it. If you're already married to a human, your human spouse will feel jealous of the computer. Your relationship with your human spouse will worsen, and might even end in divorce.

Last summer, several women got divorced, because they took my computer course. Their husbands had three complaints:

"You spend most of your time with the computer instead of with me."
"When you do spend time with me, all you want to talk about is the computer!"
"You're not the same woman I married."

To prevent such a marital problem, coax your spouse into playing a video game on the computer. If you're lucky, your spouse will enjoy the game, become hooked on it, and become as addicted to computers as you. Then your spouse will *enjoy* blabbing about computers with you and will even help you spend money on your habit.



WHY BUY A COMPUTER?

The average American has three goals: to make money, have fun, and "become a better person". Making money is called *business*; having fun is called *pleasure*; and becoming a better person is called *personal development*. The computer will help you meet all three goals: the computer will improve your business, increase your pleasure, and help you to grow into a better person.

The average sucker who buys a computer is a male who comes out at noon. During lunch hour, he walks into a computer store and says he wants to computerize his business. He wants a computer to do his accounting and also handle his mailing list. The computer store's salesman talks him into also wanting a word-processing program, to help handle business correspondence. After the sucker visits one or two other computer stores, he comes back and purchases a computer. Although the computer costs a lot, the salesman reminds him that Uncle Sam will give a tax break for buying it, since it's a "business expense".

He brings it home, but starts feeling guilty about having spent so much money. How will he convince his wife that the purchase was worthwhile?

Let's suppose he's a typical stupid male, who lives in a typically stupid house, where his typically stupid wife spends most of her typically stupid day doing typically stupid "homemaking", especially cooking. Since this is an old-fashioned marriage, he tries to convince her that the computer will help her cook. "It will help you store your recipes, darling", he coos.

"No thanks", she says. "When I find a recipe in the newspaper, I don't want to have to spend 15 minutes typing the entire recipe into the computer. I'd rather just clip the recipe out of the newspaper and tape it to a file card, which takes just a few seconds. My manual system is faster than a computerized one!"

He tries again. "You could use the computer to store your phone numbers. Whenever you have to look up a phone number, the computer will tell you instantly."

She retorts, "No thanks. To make a phone call, I don't want to have to turn on the computer, find the disk that contains the 'telephone' program, put that disk into the computer's 'disk drive', wait for the computer to ask me whose number I'm interested in, sit at the keyboard and type in the jerk's entire name, and then wait for the computer to respond. Instead of doing all that, it's quicker to just open my little black personal phone book, flip to the page where the number is, and dial my friend. Try again, lover-boy!"

"Well, darling, you could use the computer to remind you of birthdays and appointments."

"No thanks, I remember them quite well *without* a computer. And if I'm ever afraid of forgetting, I just scribble a note on my calendar, which serves just fine and costs just $3 instead of $3000. I understand that a disorganized bird-brain male, like *you*, might need a computer in order to survive; but because we women are better organized, we don't need to rely on mechanical help."

Clearly, the male is making no progress in this conversation. The truth of the matter is: the computer does *not* fulfill any real need in the home. Nevertheless, the male wants to buy it—for a variety of psychological reasons—and searches for an *excuse* to justify the purchase. In other words, the computer is a solution looking for a problem.

Of course, women buy computers also. Apple Computer Company ran a television ad that showed Dick Cavett in a kitchen, interviewing a woman who had bought a computer. "Oh, you're using it to store your recipes?" he asks.

"No!" she barks. "I'm using it to chart stocks!"

If you buy a computer, the idea of "using the computer to run your business" and "using the computer to store recipes" are both just excuses. They aren't the *real* reason you buy the computer. You *really* buy it from saying to yourself:

"I want to buy a computer because I want to be part of the twentieth century. The whole world's becoming computerized; I want to understand this new, fantastic world and get my share of the excitement.

"If I don't learn how to master the computer, the computer will master me; I'll become its slave! I might even lose my job! If I learn about computers, I can keep my job or even get a better one! Maybe I can even become a rich computer consultant!

"During the past few years, my life's become monotonous. I need a new, exciting hobby, such as a home computer. The computer will be something to continue playing with even when I reach my golden years.

"I've worked hard all my life; I deserve a computer! I'd love to own that little machine, which will obey all my commands.

"I'm tired of being intimidated by all that computer jargon. I want to learn that jargon myself; then I can intimidate my neighbors and competitors, and they'll all admire me for my brilliance!

"A computer's the only status symbol left for me to buy. Since nobody admires big luxury cars anymore, I'll have to buy a computer instead.

"The computer's a challenge. If I can master the computer, I know I'm not as stupid as people say!

"The computer's the only thing in our society that has lasting value: everything else is just a fad that comes and goes.

"Computers are exciting. They're science-fiction come true! Programming the computer is the next best thing to being an astronaut!

"The computer's a fascinating hobby that I can use in my business and even turn into an extra business on the side!"

Another reason for buying a computer is your kids. You say to yourself:

"I want my kids to grow into successful adults in the twenty-first century. To do that, they'll have to learn about computers—because computers are taking over the world!

"Already, our town's schools are requiring kids to learn about computers. If I buy my kids a computer for use at home, they'll get curious about the computer, play with it, become familiar with how it works, learn to love it, become intellectual programmers, do better in school, and grow up to become computer experts and make lots of money.

"Learning how to program the computer is better for my kids than letting them waste their lives on sex, drugs, and rock-and-roll. And it's better than letting them pop quarters into video arcades. Even if they start off by using the home computer just for games, eventually they'll get curious about how the computer's programmed, and become intellectuals."

Will the computer fulfill all those dreams? Maybe so. . . or maybe not. By the end of this book, you'll find out!

SIZES

MAXI, MINI, MICRO

Computers, like people, come in all sizes: you can get big ones, little ones, and pip-squeaks. The big'uns are called *maxicomputers* (or *mainframes*); little'uns are called *minicomputers*; and the teeny-weeny itty-bitty pip-squeaks are called *microcomputers*.

But how big is big? How little is little? How teeny-weeny is teeny-weeny? Opinions differ. It's like trying to measure a person. What's a "big" person? It could mean three things:

You're tall.
You're fat.
You're sexually well-endowed.

The same is true for computers.

Yes, the word "maxicomputer", like the word "big", is vague. But here are some guidelines. . . .

Usually, a maxicomputer fills a room; a minicomputer fits in a corner; a microcomputer freaks-out on a desktop. Usually, a maxicomputer costs anywhere from $10,000,000 down to $100,000; a minicomputer costs from $100,000 down to $10,000; a microcomputer costs from $10,000 down to $1.

Most people can't afford to spend $100,000 for a computer. In fact, most people can't afford to spend even $10,000. So most people buy *micro*computers. This year, if you're lucky, you'll even get a microcomputer as a birthday present, from your Mommy or Daddy or Hubby or Wifey.

This Guide explains how to master *all* sizes of computers—from the very biggest maxicomputers to the very smallest microcomputers. It turns you into a true virtuoso.

BORDERLINE COMPUTERS

Yes, the words "maxicomputer", "minicomputer", and "microcomputer" are vague.

For example, a computer that costs about $100,000—which is the borderline between "maxicomputer" and "minicomputer"—can be called either a "maxicomputer that's somewhat small" or a "minicomputer that's somewhat large". Your salesman will elaborate, of course: he'll call it either "the world's least expensive maxicomputer" or "the world's most powerful minicomputer" or—better yet—"the only computer that gives maxicomputer power at a minicomputer price".

Similarly, if a computer costs about $10,000, the salesman will try to convince you that it's the only computer that gives "minicomputer power at a microcomputer price".

Now that you know how to be a computer salesman, go sell lots of computers and make lots of money (and give me a cut).

WHICH COMPUTER'S BEST?

Women have different tastes, when they choose men. Some women like men who are tall. Other women prefer men who are more on-the-level.

The same goes for computers. Some people like computers that are huge. Other people prefer computers that are cuddly.

The most popular maxicomputers are made by IBM; the most popular minicomputers are made by Digital Equipment Corporation (fondly called DEC); the most popular microcomputers are made by companies such as Radio Shack. Big IBM operates out of plush offices. DEC operates out of an abandoned woolen mill. Radio Shack operates out of el-cheapo discount stores. (If you can't afford a computer, Radio Shack will try to sell you, as a booby prize, a CB radio instead.)

About computers, people get emotional. Insulting somebody's computer is like insulting your mother. People who own big IBM's say, "All those little computers are shit." People who own DEC's say, "We got a good deal." People who own Radio Shack's say, "You stupid rich bastards, our computers are's good as yours and'll take on you big boobs any day, c'mon and fight!"

But before you buy a computer, look at the competition. IBM, DEC, and Radio Shack aren't the only fish in the sea; some of their competitors are better. Treat a computer company just as you'd treat a person of the opposite sex: walk up and say, "Show me yours!"

TOP DOGS

FAVORED FIVE

The computer industry is dominated by five manufacturers: IBM, DEC, Radio Shack, Apple, and Commodore. Let's look at those favored five and their grumpy competitors.

IBM

IBM makes computers in all three sizes: it makes maxicomputers, minicomputers, and microcomputers.

IBM's *micro*computer is called the *IBM Personal Computer* (or *IBM PC*). In addition to the plain IBM PC, IBM sells a souped-up version (called the *IBM PC Extended* or *IBM PC XT*) and a stripped-down version (which is called the *IBM PC Junior* but nicknamed the *Peanut*).

IBM's main *mini*computers are the *System 34*, *System 36*, and *System 38*.

IBM's *maxi*computer used to be called the *IBM 360*. Then IBM invented an improved version, called the *IBM 370*. Today, the IBM 360 and IBM 370 are obsolete: IBM has replaced them by even newer computers, called the *3080 series* and the *4300 series*. The 3080 series comes in three versions (the 3081, 3083, and 3084); the 4300 series comes in three versions also (the 4321, 4331, and 4341).

Who competes against IBM? For microcomputers, IBM's main competitors are Radio Shack, Apple, and Commodore. For minicomputers, IBM's main competitor is DEC. For maxicomputers, IBM's main competitors are Burroughs, Honeywell, NCR, Control Data Corporation (CDC), and the Univac division of Sperry; but those competitors are all boring, and IBM is clearly the maxicomputer leader; in fact, IBM sells more maxicomputers than all its competitors combined!

Some tiny companies make computers that imitate IBM's but cost less. Their imitations are called *clones* or *plug-compatibles*. Clones of IBM's microcomputers are made by Compaq, Columbia, Corona, Eagle, and Seequa; Seequa's clone is called the *Chameleon*. Clones of IBM's maxicomputers are made by Amdahl, Nixdorf, IPL, Cambex, and National Advanced Systems (NAS).

DEC

DEC makes mainly minicomputers. DEC's most famous minicomputer is the *PDP-11*, but DEC also sells a souped-up version, called the *VAX*. The VAX is an amazingly nice computer: it's much more pleasant to program than any of IBM's minicomputers or maxicomputers.

In addition to the PDP-11 and VAX minicomputers, DEC also makes nice microcomputers and maxicomputers and tries to sell them, but few companies buy them: most companies buy microcomputers and maxicomputers from IBM instead.

DEC's amazing strength in minicomputers, and its forays into the microcomputer and maxicomputer markets also, make the company's total income quite high. DEC's total income is second only to IBM's.

DEC makes mainly minicomputers—and so do these competitors: Data General, Prime, Wang, and Hewlett-Packard.

RADIO SHACK

Radio Shack doesn't sell maxicomputers or minicomputers: it sells just microcomputers.

Radio Shack's main microcomputer is called the *Model 4*, but Radio Shack sells many other models also. In fact, Radio Shack sells a greater variety of microcomputers than any other manufacturer! It sells the Model 4, Model 4P, Model 12, Model 16B, Model 100, PC-2, PC-3, PC-4, MC-10, and Color Computer 2. To sell all those varieties, Radio Shack has 8,000 stores.

Before inventing those varieties, Radio Shack invented several older varieties: the Model 1, Model 2, Model 3, Model 16, PC-1, and Color Computer 1. Although the older varieties are obsolete, many people continue to use them.

Since Radio Shack is owned by Tandy Corp., it's called *Tandy's Radio Shack* (or *TRS*). Radio Shack's entire computer line is called the *TRS-80*.

APPLE

Apple Computer Corp. sells just microcomputers.

Its first computer was the Apple 1, then came the Apple 2, then the Apple 2-plus, and finally the *Apple 2e*. The Apple 1, Apple 2, Apple 2-plus, and Apple 2e are resemble each other. In fact, programs written for the Apple 1, Apple 2, and Apple 2-plus still work on the Apple 2e. Today Apple sells just the Apple 2e, but many people still own the earlier models.

Apple also sells two fancier models (the *Apple 3* and the *Lisa*), but their higher prices make them less popular. Lately, Apple's been developing a further improvement, called the *Macintosh*.

The Apple 2e is not as modern as the latest microcomputers from Apple's competitors (such as IBM, Radio Shack, and Commodore). Compared to those competitors, the Apple 2e is primitive, old-fashioned, and overpriced. Nevertheless, many people buy the Apple 2e, because more programs work on the Apple 2e than on any other computer. That's because the Apple 2e runs all the programs that were written for the Apple 2-plus, and also all the programs that were written for the Apple 2, and even all the programs that were written for the Apple 1.

Many small companies make computers that imitate the Apple but cost less. Those imitations are called *Apple clones*. For example, you can buy an imitation called the *Pineapple*; another company makes an imitation called the *Pear*; another clone's called the *Orange*; another's called the *Micro Professor*; another's called the *Franklin Ace* (because it appeals to thrifty people like Benjamin Franklin); and Coleco makes an Apple clone called the *Adam*.

Apple Computer has sued most of the companies that make Apple clones, and so most of the clones are not available in the United States: you must buy them overseas. The only clones still being marketed in the United States are the Franklin Ace (because its company has a clever lawyer), the Orange (which is different enough from the Apple to avoid legal problems), and Coleco's Adam (which does only part of what the Apple can).

COMMODORE

Commodore manufactures just microcomputers. Its two most popular microcomputers are the *VIC-20* and the *Commodore 64*. They're inexpensive: you can buy the VIC-20 for just $85, and you can buy the Commodore 64 for just $200. To get those low prices, shop at these discount department stores: Child World, Toys R Us, and K Mart.

Because of those low prices, the VIC-20 and Commodore 64 are more popular than the computers sold by Radio Shack and IBM—even though Radio Shack's and IBM's computers are better. If you're rich, buy a Radio Shack or an IBM; if you're poor, buy a VIC-20 or a Commodore 64.

Back in 1982, when the VIC-20 and Commodore 64 were slightly more expensive, Commodore's main competitors were Texas Instruments, Atari, and Timex Sinclair. But in 1983, Commodore reduced its prices; Commodore's lowered prices are driving Texas Instruments, Atari, and Timex Sinclair out of business. The three losers—Texas Instruments, Atari, and Timex Sinclair—are trying to launch a counter-attack, but I doubt the counter-attack will succeed. So far, the three losers have been running in the red, to the tune of many millions of dollars!

Besides the VIC-20 and the Commodore 64, Commodore also makes some computers that are slightly more expensive. But if you're rich enough to afford Commodore's expensive computers, you should continue saving your pennies until you can afford a Radio Shack or Apple or IBM instead, since Commodore's computers are inferior.

CP/M COMPUTERS

The computer industry is dominated by the five companies I've discussed (IBM, DEC, Radio Shack, Apple, and Commodore) and also by a sixth force, called the *CP/M cartel*.

The CP/M cartel is a group of small companies that have banded together and have produced computers that resemble each other and can share each other's programs. The most famous companies that belong to the CP/M cartel are Osborne and Kaypro. Osborne recently went bankrupt, but Kaypro is doing quite well. In addition to Osborne and Kaypro, the group consists of hundreds of other companies also, such as Xerox, Morrow, Cromemco, Vector, North Star, and Zorba.

Because the IBM Personal Computer has become so popular, programmers who wrote programs for the CP/M cartel are now rewriting the programs so that they'll also work on the IBM Personal Computer. The CP/M cartel's influence is waning, as more programmers and users are switching to the IBM PC. Many members of the CP/M cartel are "giving up", and plan to start selling IBM PC clones instead.

For example, Kaypro still sells its CP/M computer, but also sells a board full of electronics to insert into the computer, to turn it into an IBM PC clone.

EQUIPMENT

THE TYPICAL MICROCOMPUTER

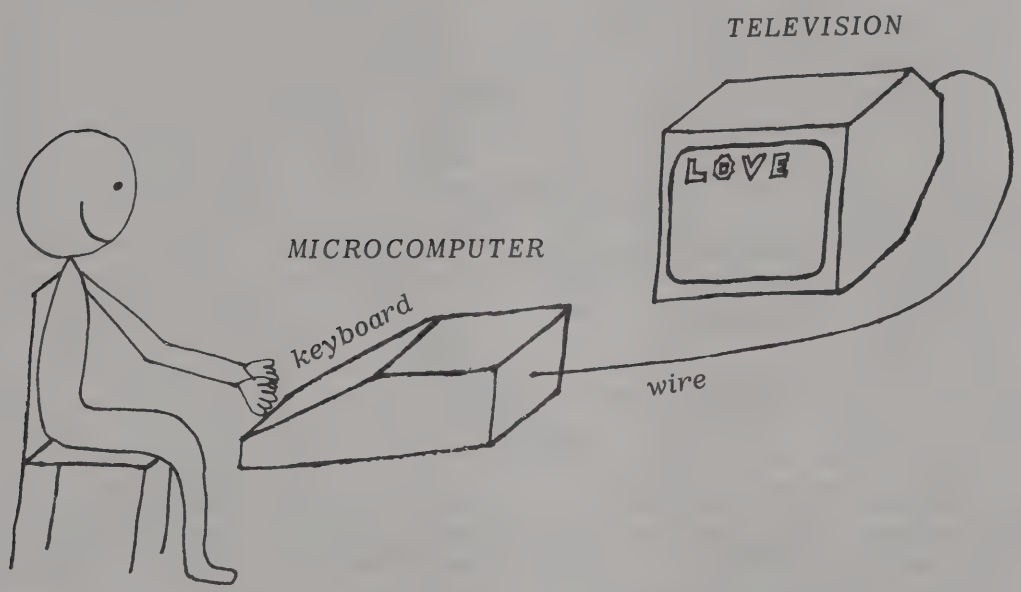
Computers come in all shapes and sizes. I'll begin by explaining the typical microcomputer. Later, I'll explain *your* computer.

The typical microcomputer is a little box. It's about a foot long, a foot wide, and three inches high. Like a portable typewriter, the top of the box is slightly sloped and has keys on it. The keys contain all the letters of the alphabet, all the digits (from 0 to 9), punctuation marks (period, comma, colon, semicolon, apostrophe, quotation mark, and parentheses), math operations (plus, minus, and symbols for multiplication and division), and some extra symbols also.

A typewriter uses paper, but a typical microcomputer does not. Instead of using paper, the microcomputer uses a television set.

A wire runs from the microcomputer to your home's TV set. The microcomputer turns you into a TV star; whatever you type appears on TV!

For example, if you type the word LOVE, you see the word LOVE on your TV, like this:



RCA CORD

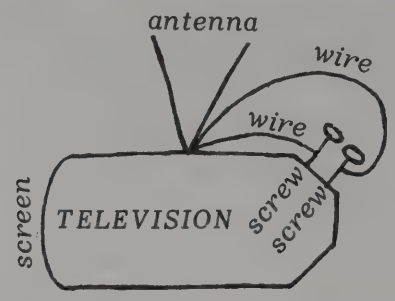
The "wire" that runs from the microcomputer to the TV is actually a *bundle* of wires, and is therefore called a *cord*.

Most microcomputers use a kind of cord invented by RCA. It's called an *RCA cord* and looks like this:

HOW TO PREPARE YOUR TV

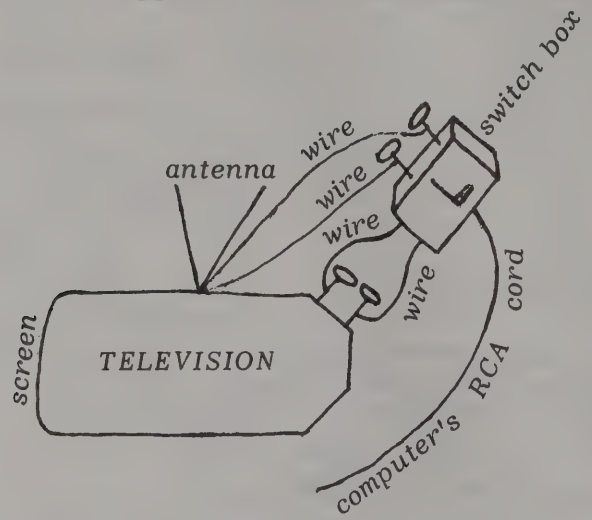
You must prepare your TV, to attach it to your microcomputer. Here's how.

Look at your TV antenna. Wires run from the antenna to two screws, which are on the back of the TV:



Loosen the screws, to release the antenna. Then—instead of attaching the antenna—attach a special box, called a *switch box*. Attach the switch box to your TV's screws.

Plug the computer's RCA cord into one side of the switch box. Attach the antenna's wires to the opposite side of the switch box:



The switch box has a switch on it. If you move the switch towards the antenna, you have a normal TV, so you can watch Johnny Carson. If you move the switch towards the computer's RCA cord, your TV's controlled by the computer, so that the computer can write messages such as LOVE on your TV screen.

So by moving the switch, you can make your TV act either normally or computerized. Your family will argue about which way to move the switch.

That switch box is the same kind used by video games (such as Mattel Intellivision, Atari VCS, and Coleco). When you buy the computer, the salesman will give you the switch box and RCA cord, free!

To use the computer, you must turn your TV's dial to the correct channel. Most computers (by Commodore, Texas Instruments, and Radio Shack) use channels 3 and 4.

To get a sharp picture on your TV screen, *avoid* the channel used by your local TV station. For example, if you live in Boston or New York City, NBC hogs channel 4, so *avoid* channel 4: put your computer on channel 3 instead. (Turn your TV's dial to channel 3, and also turn the computer's switch to channel 3.)

Although most computers use channels 3 and 4, some computers use different channels instead. For example, Atari and Timex Sinclair use channels 2 and 3 (instead of "3 and 4"). High-priced computers (by Apple and IBM) use higher channel numbers instead: 10 and 33 and 34.

If the image on your TV screen looks fuzzy—so that you can barely read the computer's writing—adjust the TV's "fine tuning" knob.

HOW TO SOLVE PROBLEMS

To make the computer solve a problem and obey you, type a command on the computer's keyboard. The computer will obey the command.

The TV screen will show the command you typed and also show the computer's response. Hopefully, the computer's response is the solution to the problem you were trying to solve!

RAM

The computer contains a lot of electronic circuitry. That circuitry's hidden inside the computer.

One part of the circuitry is called the *random-access memory (RAM)*. The RAM's main purpose is to remember what problem you fed the computer.

Unfortunately, the RAM requires electricity. If the RAM loses electricity (because you pulled out the plug, or because a battery wore out, or because your town is having a blackout), it forgets what problem you fed the computer, and the computer doesn't know what to do!

TAPE

Suppose you're working late at night (with the help of your computer) and become tired, but you and the computer haven't finished the work yet. You want to go to bed, and continue the next morning where you left off.

To save electricity, you want to turn off the computer's power for the night. But if you turn off the computer's power, the RAM will forget what problem you've been working on, and won't be able to continue the work the next morning.

Here's a trick for making the computer continue the work the next morning, even if you turn off the power overnight. . . .

Get a cassette tape recorder—the kind that plays music. (If you don't own a cassette tape recorder already, borrow one from your neighborhood's punk teenager.) At night, before you go to bed, make the computer sing a song to the tape recorder. The song is a special code that stands for the information in the RAM. The tape recorder puts the song onto the tape. Then turn off the computer and the tape recorder, and go to bed.

The next morning, when you wake up, turn on the computer and the tape recorder, and play back the tape. The computer hears the song on the tape and realizes that the song's a code for what to put in the RAM. The computer listens to the song, obeys the code, and puts the RAM back to the same condition as the previous night. Then you can continue working on your problem, exactly where you left off.

To make all that happen, you must connect the computer to the tape recorder. Here's how. . . .

In the side of the tape recorder, you'll see holes for attaching earplugs and a microphone. Those holes are marked "earphone", "microphone", "remote", and "auxiliary". Run wires from your computer into those holes, so that the computer pretends to be the tape recorder's earphone and microphone.

To make the computer sing a song to the tape recorder, press the buttons on the tape recorder marked "RECORD" and "PLAY", then type a special command on the computer's keyboard. In the morning, to make the computer listen to that tape, type another special command on the computer's keyboard and then press the tape recorder's "PLAY" button.

So to store information overnight (while the computer's shut off), store the information on tape.

If you've stored information on tape, you can mail the tape to your friend. Your friend can put the tape in his own tape recorder, attach his tape recorder to his computer, and make his computer listen to your tape. Then his computer will act like yours: it will get the same information into its RAM.

You can buy tapes that will put useful information into your RAM. For example, you can buy tapes that will help your computer analyze statistics, do word processing, create poetry, and perform other exciting feats.

DISKS

Although tape works okay, it's somewhat slow, clumsy, and unreliable. Instead of using tape, you should use a *disk*.

You can buy many kinds of disks. The disk's diameter can be 3 inches, 5 inches, 8 inches, or 14 inches. The material the disk is maue of can be thick or thin. If the material is thick, the disk is called *hard*, because the disk is hard to bend. If the material is thin, the disk is called *floppy*, because the disk bends easily; but do *not* bend it; bending the disk hurts it.

The cheapest kind of disk is the *5-inch floppy disk*. Because it's the cheapest, it's the most popular (even though it's crummy). Since its diameter's about 5 inches, it looks like a 45 RPM record. Since it's floppy, it bends easily, and you must be careful *not* to bend it.

The disk acts like a tape: the computer sings a song onto the disk; the song's a code that stands for information in the RAM. Like a tape, the disk can be erased and then re-used. The disk is better than a tape, because the disk is faster, more convenient, and more reliable.

The disk is round. If the disk is floppy, it comes in a square black jacket, which helps protect the disk from bending and from dust. The jacket stays on the disk permanently. The jacket has a hole in it: computer equipment reads the information on the disk by peeking through the hole.

To use the disk, you put the disk (including its jacket) into a *disk drive*, which resembles a record player. The disk drive is a box that has a turntable hidden inside.

The disk drive (which is a box) has a slit in its front side. To use the disk drive, you slip the disk (including its jacket) into the slit. (Putting your sheathed treasure into the box's slit can feel quite sexy. But don't shove too hard. Please be gentle!)

After putting your disk into the slit, you cover the slit by pulling shut the slit's little *door*.

When the disk drive realizes you've shut its door, it knows you've inserted the disk, and so it automatically positions your disk onto the turntable. The turntable's called a *spindle*; you never see it, because it's hidden inside the disk drive.

So to use a disk, you must buy a disk drive. Unfortunately, the disk drive's expensive. It costs much more than a cassette tape recorder. A cassette tape recorder usually costs between $20 and $75; a disk drive usually costs between $200 and $600.

If you have very little money, buy a cassette tape recorder (because it costs less than a disk drive); if you're richer, buy a disk drive (because it works better than a tape recorder). For "fooling around in your home", a tape recorder is good enough; but if you want the computer to run your *business*, you'd better buy a disk drive.

If you and your friend are both rich enough to have disk drives, and you've stored information on your own disk, you can mail the disk to your friend, so your friend can feed your information into his computer.

You can buy disks that contain information to help the computer do word processing, filing, accounting, tutoring, and other fancy tasks.

SCREENS

The typical microcomputer attaches to a TV. Unfortunately, the image on the TV's screen is fuzzy. For a better, sharper picture, get a *computer monitor* instead of a TV.

Like a TV, the monitor has a picture tube, whose front surface is the screen you watch. The technical name for "picture tube" is *cathode-ray tube (CRT)*.

Like a TV, the monitor attaches to a typical microcomputer by using an RCA cord. Attaching the monitor's even easier than attaching a TV.

To attach a microcomputer to a TV, you must put a switchbox on the back of the TV, and then plug the RCA cord into the switchbox. The monitor's easier than a TV, because the monitor doesn't require a switchbox: the RCA cord plugs directly into a special hole in the back of the monitor.

When you buy a TV, you ask for either "color" or "black-and-white". Similarly, when you buy a computer monitor, you ask for either *color* or *monochrome*. A color monitor displays all the colors of the rainbow; a monochrome monitor displays just dark-and-light.

The typical monochrome screen displays black-and-white, but fancier monochrome screens look tinted. For example, you can buy a *green-screen monitor*, which displays dark green instead of black, and displays light green instead of white. You can also buy an *amber-screen monitor*, which displays dark orange instead of black, and displays light orange instead of white. Green-screen and amber-screen monitors are easier on your eyes than black-and-white monitors, because black-and-white monitors tend to produce too much glare.

In 1960 the Pickett Slide Rule Company did research to find out which monochrome color is easiest on the eye, and concluded that the best color is "yellow with a slight tinge of green". So they manufactured slide rules that were greenish yellow. But unfortunately, computer companies are stupid, don't know about Pickett's research, and don't sell greenish yellow monitors.

A color TV costs more than black-and-white. Similarly, a color monitor (that displays all the colors of the rainbow) costs more than a monochrome monitor.

The typical monochrome monitor costs between $100 and $150. The typical color monitor costs between $550 and $700. (If you pay less than $550 for a color monitor, you get junk.) Those prices are for monitors that have 12-inch or 13-inch screens.

Monitors cost more than TV's. If you go to a discount department store, you can get a TV cheaply: a 12-inch black-and-white for $60-$80, a 13-inch color for $225-$250, and a 19-inch color for $300-$350.

So a monitor's better than a TV but costs more. Is it worth the extra cost? Not necessarily.

If your computer's cheap and intended for cheap people who have cheap TV's, it purposely makes the letters on the TV gigantic, so you can read them easily. (The letters are big, garish, and ugly, but you can read them!) For such a computer, don't bother buying a monitor: it's an unnecessary expense. Buying a monitor for a cheap computer is as silly as buying caviar for a five-year-old kid whose stomach was intended for hamburgers.

On the other hand, if your computer makes the letters smaller and finer (so that more letters can fit on the screen), a monitor can help clarify the computer's beauty.

Before you start debating between a TV and a monitor, check your computer. Some computers attach to TV's but not to monitors; some computers attach to monitors but not TV's; some computers can attach to either and give you a choice.

Some computers come with a monitor already attached, permanently. That's called a *built-in monitor*. Instead of using an RCA cord, a built-in monitor uses a cord that's permanently attached to the monitor and to the computer's circuitry; the cord is hidden from view.

If your computer's large, the computer's in a big box that has no keyboard and no screen. A wire runs from that big box to a *video-display terminal (VDT)*. That terminal is a keyboard-screen combination: it's a keyboard with a built-in monitor.

When you travel, you'll want to bring your computer with you. Wouldn't it be nice to have a computer system small enough to fit in your suitcase—or even in your pocket?

Unfortunately, a TV or monitor contains a picture tube, which is heavy and bulky. It just barely fits in a suitcase, and it certainly can*not* fit in your pocket!

Today you can buy a pocket-size computer. Instead of using a picture tube, it uses a *liquid-crystal display (LCD)*. The LCD is smaller than a picture tube; it also weighs less; and it consumes so little electricity that it can run on batteries! The LCD produces tiny black letters on a white background.

Look at your arm. If you're wearing a digital watch, look at the tiny screen that displays the time. That screen's an LCD, controlled by a miniature computer inside the watch! The screen is tiny, light-weight, and runs on batteries.

Look at your pocket calculator. If its screen displays black numbers on a white background, the screen's an LCD.

Today you can buy a pocket-size computer that looks like a pocket calculator but is much more intelligent. The pocket-size computer's keyboard includes all the letters of the alphabet: you can type English words on the keyboard, and the computer understands them.

TYPES OF COMPUTERS

You can buy five types of computers. The cheapest is called "type 0"; the most expensive is called "type 4". Let's look at them.

A normal computer has a keyboard containing all the letters of the alphabet. If your computer is so crummy and stripped-down that it does *not* have a keyboard containing all the letters of the alphabet, your computer's called "type 0", which means "it's the pits!" A type 0 computer hides inside your pocket calculator and inside your digital watch. You can also find it inside video games: you can find it inside the big arcade games, the home video-game machines such as Mattel's Intellivision, and the hand-held electronic games such as Merlin). You can also find it inside some "computerized" electronic appliances in your home: it hides inside the appliances and is never seen. The typical type 0 computer costs the manufacturer between $1 and $50.

A type 1 computer includes a keyboard that contains all the letters of the alphabet; it also includes an LCD screen. It's better than a type 0 computer, because it has a better keyboard. Since the LCD screen is small, light-weight, and runs on batteries, this type of computer is portable. Its portability appeals to traveling executives and to people who work outdoors. The most popular type 1 computers are sold by Radio Shack and are called the Pocket Computer 4 ($70), the Pocket Computer 3 ($100), the Pocket Computer 2 ($200), and the TRS-80 model 100 (which is much fancier and costs $800).

A type 2 computer uses a TV. Since a TV is larger (and displays more information) than an LCD screen, and since the average American home contains a TV already, this type of computer appeals to Americans who want to use the computer at home. The typical American who uses a type 2 computer attaches it to a cassette tape recorder. Here are the most popular type 2 computers and their approximate prices from discount dealers: Timex Sinclair 1000 ($35), Texas Instruments 99/4A ($60), Commodore VIC-20 ($85), Radio Shack Color Computer 2 ($150), Atari 600 XL ($150), Commodore 64 ($200), and Coleco Adam ($600).

A type 3 computer uses a monitor and displays 80 characters per line. If a computer uses a monitor but displays slightly less than 80 characters per line, it's "almost" a type 3 computer—and it's obsolescent. For example, Radio Shack replaced its TRS-80 model 3 computer (which displayed 64 characters per line) by the model 4 computer (which displays 80 characters per line). Osborne Computer Company replaced its Osborne 1 computer (which displayed 52 characters per line) by the Osborne Executive Computer (which displays 80 characters per line); but the company went bankrupt anyway. Apple Computer Company encourages its customers to buy an "80-column card", which changes the Apple 2e computer from 40 characters per line to 80 character per line.

The most popular manufacturers of type 3 computers are Radio Shack (which makes the TRS-80 model 4 and TRS-80 model 12 and TRS-80 model 16), Apple (which makes the Apple 2e, Apple 3, and Lisa), Franklin (which makes the Ace 1000 and Ace 1200, which imitate the Apple), Kaypro (which makes the Kaypro 2, Kaypro 4, and Kaypro 10), and IBM (which makes the IBM Personal Computer). In addition to those manufacturers, there are *hundreds* of others: shop around before you buy!

Type 3 computers appeal to managers of small businesses. The typical manager who buys this type of computer gets two disk drives. The total cost for a type 3 computer, including a monitor and two disk drives, is usually between $1500 and $3500. The exact price depends on which brand you buy and whether you go to a discount dealer.

When you buy a type 3 computer, you're supposed to use a monitor and disks. But some type 3 computers are flexible: they let you replace the monitor by a TV, and let you replace the disk drive by a cassette tape recorder, to reduce your cost. But if you try to save money by making those substitutions, your computer won't work as well. For example, if you replace the monitor by a TV, you'll see only 40 characters per line.

A type 4 computer handles several terminals simultaneously. Each terminal has a person sitting at it; so the computer is making many people happy at once! The most popular type 4 computers are manufactured by IBM and DEC, but you can choose from hundreds of other manufacturers also. The typical type 4 computer costs between $8,000 and $10,000,000. Those computers use extra-large disks and extra-large tapes.

PRINTERS

The typical computer writes its answers on a screen. The screen is either a TV or a monitor or an LCD.

If you want the answers to be on paper instead, you must buy a *printer*, which is a device that prints on paper. A cable of wires runs from the printer to the computer.

The typical printer looks like a typewriter but doesn't necessarily have a keyboard. If has a keyboard, it's called a *keyboard send-receive printer (KSR printer)* or a *printing terminal* or a *hard-copy terminal*. If it lacks a keyboard, it's called a *receive-only printer (RO printer)*.

You'll probably buy a printer that *lacks* a keyboard, because your computer probably came with an adequate keyboard already.

Printers come in all sizes and prices. The cheapest is Radio Shack's PC-4 printer, which costs just $80 and prints slowly: just 20 characters per second. The most expensive is the laser printer (sold by IBM and Honeywell), which costs hundreds of thousands of dollars and prints very quickly: 5 pages per second!

The most popular printers are compromises: they cost between $500 and $1500, and they print between 25 and 100 characters per second.

Summary: after buying a computer, you'll probably also buy a printer that costs between $500 and $1500.

SOFTWARE

COMPUTER LANGUAGES

To give the computer a command, you sit at the computer's keyboard and type the command in English. Computers around the world understand English (rather than Chinese or Russian or Spanish or French), because English is the international language of science.

Because the English language is so complicated and so vast, no computer understands *all* of English. Instead, each computer understands only *part* of English. The computer understands only a few words, a few phrases. The words and phrases that the computer understands are called the *computer's language*. The computer's language is a small part of English.

Different computers understand different parts of English. The most popular part of English is called *BASIC*. It consists of words such as PRINT, GO, TO, INPUT, IF, and THEN. Most computers understands BASIC. So to make the computer do what you wish, you put your fingers on the keyboard and type a command that contains the words PRINT, GO, TO, INPUT, IF, and THEN.

How does the computer know the meaning of the words PRINT, GO, TO, INPUT, IF, and THEN? Somehow, the definitions of those words must be in the computer's memory.

The definitions can get into the memory two ways. One way is to buy a disk that contains those definitions, and then tell the computer to copy those definitions from the disk to the RAM. The other way is to put inside the computer a hidden electronic circuit, called a *read-only memory (ROM)*, in which those definitions are permanently engraved.

Although the most popular part of English is BASIC (which consists of words such as PRINT, GO, TO, INPUT, IF, and THEN), some computers use a different part of English instead. One part of English is called PASCAL; another part's called COBOL; another part's called FORTRAN. Each of those parts—BASIC, PASCAL, COBOL, and FORTRAN—is called a *computer language*. Each computer language is part of English. In addition to BASIC, PASCAL, COBOL, and FORTRAN, there are *hundreds* of other computer languages.

When you're using a computer, you use just one computer language at a time. If you wish to switch to a second computer language, you must feed the computer the second language's definitions, by inserting a second disk or a second ROM.

WHAT'S A PROGRAM?

To make the computer accomplish a complicated task (such as your company's payroll), you give the computer a list of commands. Each command must be written in a computer language (such as BASIC). The entire list of commands needed to accomplish the task is called the *program*.

For example, you can write a program to make the computer do the payroll. You can write a program to make the computer play chess. You can write a program to make the computer imitate a psychotherapist.

If every command in the program is written in BASIC, the program is called a *BASIC program*. (If every command in the program is written in PASCAL, the program is called a *PASCAL program*.)

To feed the program into the computer, sit at the computer's keyboard, and type the program. The computer automatically puts the program into the computer's RAM. After you've typed the entire program (so that the entire program is in the RAM), you can type a special command that makes the computer copy the program from the RAM to a disk (or tape). The disk (or tape) will retain the program even if you turn off the computer's power.

If you're too lazy to invent a program, you can buy one. For example, you can buy a program that will make the computer do your company's payroll. If you go into a computer store and say that you want to buy a program, the salesman will hand you a disk (or tape) that contains the program on it already.

To use a disk, you put the disk into the disk drive, sit at the keyboard, and type special commands that tell the computer to use the disk. The commands you type are called *disk commands*.

But how does the computer know what those disk commands mean? Before giving a disk command, you must make sure the computer has been fed the disk command's definition.

When you buy a computer with a disk drive, the salesman will give you a special disk to put into the disk drive. That special disk, which is called the *system master*, contains the definitions of all the disk commands.

When you turn on the computer, the computer examines that system-master disk, finds the disk-command definitions on it, and copies those definitions from the disk to the RAM. Later, if you ever give a disk command, the computer looks at the RAM to find the command's definition and then obeys the command.

The collection of disk-command definitions is called the *disk operating system*. The abbreviation for disk operating system is *DOS*, which is pronounced "doss". The system master contains the DOS; when you turn the computer on, the computer copies the DOS from the disk to the RAM.

In addition to the DOS, the system master also contains some programs. Some of the programs are very helpful; they are called *utilities*. The remaining programs are just cute demonstrations that play music, show pretty colors, etc.; those programs are called *demonstration programs*.

If you buy a blank disk and insert it into the disk drive, you cannot use the disk easily, unless the disk contains at least part of the DOS. So you should tell the computer to copy part of the DOS from the RAM to the blank disk.

What commands do you type, to make the computer use a disk? The answer depends on which commands the computer understands. Different computers understand different disk commands, because different computers use different DOS's.

For example, Apple computers use *Apple DOS*; Tandy's Radio Shack computers use *TRSDOS* (which is pronounced "trissdoss"); IBM's Personal Computer uses *PC-DOS* (which is also called *MS-DOS*, which is pronounced "em ess doss").

Another popular DOS is *CP/M* (which stands for "control program for micros"). You can use CP/M on every type 3 microcomputer. (A "type 3" microcomputer is a microcomputer that uses a monitor and displays 80 characters per line.) For example, if you buy an Osborne or Kaypro microcomputer, you get CP/M free. On Apple, Radio Shack, and IBM microcomputers, you can get CP/M as an extra-cost option.

Suppose you buy a disk containing a program. Before you use that disk's program, you must put a DOS into your computer's RAM. The DOS tells the computer how to handle the disk. The DOS is the "boss"; it controls the computer's handling of the disk.

If you see an ad for a program, and the ad says the program "runs under CP/M", it means you must buy the CP/M DOS and put it into your RAM, before using the program. If you see an ad that says the program "runs under TRSDOS", it means the program requires TRSDOS instead of CP/M.

APPLICATIONS PACKAGES

The typical program accomplishes just one task (such as your payroll). But if the program is fancy, it accomplishes *many* tasks. A program that accomplishes many tasks is called an *applications package*.

A *word-processing program* is an example of an applications package. A word-processing program helps you write memos (or reports or books); it also helps you edit what you wrote. It can help you write *anything* to *anybody* about *any topic*; that's what makes it such a wonderful applications package.

The word-processing program that most people buy is called *Wordstar*; it comes on a disk and runs under CP/M. Personally, I don't like Wordstar, because it's unnecessarily complicated and overpriced. My favorite word-processing program is *Super Scripsit*, which costs less than Wordstar, is easier to learn how to use than Wordstar, and accomplishes a greater variety of tasks than Wordstar. Super Scripsit comes on a disk, costs $199, runs under TRSDOS, and works on only Radio Shack's computers: model 3 and model 4.

An *electronic spreadsheet* is a program that helps you do accounting. It can help you do any kind of accounting: it can help you do inventory, general ledger, budget planning, etc. Because it's so flexible, it's an applications package.

The first electronic spreadsheet program was called *Visicalc*. Then came an improvement called *Supercalc*. Then came a further improvement called *Multiplan*. Finally came the "ultimate" electronic spreadsheet, which is called *1-2-3*: it comes on a disk and costs $495.

Although 1-2-3 was originally written for the IBM Personal Computer, it's being rewritten for the Texas Instruments Professional Computer and for several other computers also. It runs under MS-DOS.

A *data-management system* is a program that helps you manipulate lists of facts. For example, it can help you handle a mailing list: it will print the names and addresses in alphabetical order or zip-code order or any other kind of order you like. It can replace your phone directory: it will put the phone numbers of all your friends onto a disk, and let you retrieve the phone numbers easily. It will also store whatever information you wish, about your customers, suppliers, employees, friends, enemies, students, teachers, libraries, and hobbies; it puts all that data about your business and life onto the disk, and then lets you retrieve that data easily.

The nicest data-management system is called *PFS File*. It's easy to learn how to use. It comes on a disk. It runs only on computers by Apple, Radio Shack, and IBM. (It runs on the Apple 2-plus, the Apple 2e, the Apple 3, the TRS-80 model 4, and the IBM Personal Computer.) For Apple and Radio Shack computers, PFS File costs $125; for IBM computers, PFS File costs $140.

The fanciest data-management system for microcomputers is called *Dbase 2*. It's much harder to learn how to use than PFS File; but it can accomplish a greater variety of tasks than PFS File. It comes on a disk. It's expensive: its list price is $675, though discount dealers sell it for just $420. It comes in two versions: one version runs under CP/M; the other version (which is mainly for the IBM Personal Computer) runs under MS-DOS.

Although the Apple can run Dbase 2 (if you buy CP/M for the Apple), you should avoid doing so, because Dbase 2 handles Apple's disks poorly. Instead, you should buy a different data-management system called *DB Master*, which isn't as fancy as Dbase 2 but uses Apple's disks more wisely. DB Master costs $225 and runs under Apple DOS.

A group of interrelated lists of data is called a *data base*. If a data-management system can handle a whole data base simultaneously, it's called a "data-*base*-management system" (*DBMS*). For example, Dbase 2 (which is fancy) is a DBMS; PFS File and DB Master (which are simpler) are *not* DBMS's.

Those are the most popular applications packages for business. But you can also buy applications packages for art, music, education, and many other fields.

HARDWARE VERSUS SOFTWARE

Computer equipment is called *hardware*. The hardware includes the keyboard, the screen, the disk drive (or tape recorder), the printer, and the hidden electronic circuitry.

The hardware even includes parts such as the nuts, bolts, screws, and wires. If you lose any of those parts, you can buy replacements at your local hardware store.

Information inside the computer is called *software*. It consists of programs, plus all the information the computer needs in order to handle the programs. A fancy computer's software includes brief programs but also includes applications packages, a DOS, utilities, and a language processor. (The language processor is a list of definitions of words used by a computer language.)

Usually, the software comes on a disk (or tape), and the computer copies the software from the disk to the RAM. If, instead, the software is firmly and permanently etched into the computer's ROM circuitry, it's called *firmware*.

Before you buy a computer, examine the computer's hardware and also its software.

WHAT TO BUY

Which computer should you buy? There's no simple answer.

For word processing, I recommend the Radio Shack model 4, because it runs Super Scripsit. For accounting, I recommend the IBM Personal Computer, because it runs 1-2-3. For data management (such as mailing lists and filing), I recommend the Apple or Radio Shack model 4 or IBM Personal Computer, because those computers run PFS File.

For a computer to practice programming on, I recommend the IBM Personal Computer, because it has the fanciest version of BASIC. If you find that computer too expensive, try the Radio Shack Color Computer 2 (with Extended BASIC), which is almost as good for programming but costs much less. The Radio Shack Color Computer 2 with Extended BASIC costs just $320 from Radio Shack; it costs even less if you go to a discount dealer (such as Computer Plus, 245A Great Rd., Littleton, MA 01460, phone 617-486-3193, out-of-state phone 800-343-8124). If you're so poor that you can't afford even the Radio Shack Color Computer 2, get the Commodore VIC-20, which costs just $85. If you can't afford even $85, I recommend saving your pennies until you *can*!

For a variety of fun and educational activities, I recommend the Apple 2e, because it offers the greatest variety of clever fun and educational programs.

In addition to those computers, you should also consider imitators, which offer similar abilities but charge less.

If you want to computerize *all* those applications in the best way possible, you must buy *all* those computers! If you're not that rich, choose the computer that's best at the application you care about most. Choosing a computer is like choosing a car: the best one for you depends on your personal tastes; it depends on which features are most important for your own personal needs.

Even though I like Super Scripsit and 1-2-3 and PFS File and IBM's BASIC, your own tastes might be different. Shop around; look at the competition. Walk into your local computer store and ask questions. Maybe the salesman will show you a newer and even better computer program I haven't heard of. If so, tell me, so I can put it in the next edition!

CHAPTER 2

PERSONAL PROGRAMMING

WARM-UP

WHAT'S BASIC?

To operate a computer, you put your fingers on the computer's keyboard and type commands. You type the commands in English.

The computer understands only a small part of English; it understands just a few words and phrases. The words and phrases the computer understands are called the *computer's language*.

Most computers use a language called *BASIC*, which was invented at Dartmouth College by two professors: John Kemeny and Tom Kurtz. They invented it in 1964, and continuously improved it until 1971. The 1971 version consists of words such as PRINT, GO, TO, INPUT, IF, and THEN.

This chapter and the next few chapters explain how to use those words.

MICROSOFT BASIC

Different computers speak different *dialects* of BASIC. The most popular dialect was invented in 1975 by a 19-year-old kid named Bill Gates. He called himself "Microsoft", because he developed software for microcomputers. The dialect of BASIC he invented is called *Microsoft BASIC*.

Because his dialect of BASIC is so wonderful, all the popular computer companies have paid him, to make their computers understand Microsoft BASIC. That's right: IBM, Apple, Radio Shack, Commodore, and *hundreds* of other computer companies all had to pay off Bill.

Because Microsoft BASIC's become so popular, Bill's had to hire dozens of employees to help him fill all the orders. Microsoft has become a multi-million-dollar company.

So now Bill's rich, finally retired (from the presidency of Microsoft) at the ripe old age of 27, and still single. If you're a woman who's looking for a brilliant, rich, handsome husband, maybe Bill's you're man! I'll tell you more about Bill later. (Heh, heh!)

Over the years, Bill gradually improved Microsoft BASIC. His original version of Microsoft BASIC was called "version 1". Then came an improvement, called "version 2"; then came "version 3"; then came "version 4"; then came "version 4-extended"; then came "version 5". Finally came the ultimate version, which Bill calls "version GW". The "GW" stands for "gee whiz!"

Versions 1, 2, and 3 were bad; nobody uses them anymore. Version 4 (or a slight variation of it) is still used by Commodore and by most Apple computers (the Apple 2, 2-plus, and 2e). Version 4-extended is used by the Apple 3 and by most Radio Shack computers (Radio Shack's TRS-80 models 1, 2, 3, 12, and 16). Version 5 is used by Radio Shack's newest computer (the TRS-80 model 4); it's also used by computers that have CP/M (such as Osborne and Kaypro). Version GW—the ultimate version—is used by the IBM Personal Computer.

Some computers use strange variations. For example, Radio Shack's Color Computer (with Extended BASIC) uses version 4 but also includes *some* features from version GW. The Texas Instruments 99/4A uses a version of Microsoft BASIC that's so ridiculous that Bill's ashamed to admit he had anything to do with it!

WHAT'S IN THIS BOOK?

This book explains all the major computer languages. It begins with the most popular language: BASIC. And it begins with the most popular dialect of BASIC, which is Microsoft BASIC. It begins with *Modern Microsoft BASIC* (which includes the best features of versions 4, 4-extended, 5, and GW).

If your computer doesn't understand Modern Microsoft BASIC, don't worry! At the end of this volume you'll find an amazing appendix, which reveals the intimate details of computers by Apple, Commodore, Radio Shack, Osborne, Kaypro, IBM, Texas Instruments, Sinclair, Atari, DEC, and other brands, including yours. It reveals how each brand differs slightly from Modern Microsoft BASIC.

So after reading the main text (which explains Modern Microsoft BASIC), check the appendix, to discover your own computer's peculiarities.

And remember: if you ever get confused, phone me at 617-266-8128, and I'll help you, free!

THE KEYBOARD

The computer's keyboard looks like this:

You need all those keys, for Modern Microsoft BASIC. (Just to be cute, your computer might have extra keys, and the keys might be arranged more exotically. To find out about *your* computer's keyboard, see the appendix.)

Now I'm going to explain how to use the keys. (But don't touch them, until you've read the next section, which is called GET STARTED.)

Numbers. The top row of keys contains the numbers. Don't confuse 1 with I; the number 1 is in the top row. Don't confuse zero with the letter O; the number zero is in the top row, and might have a slash through it, like this: Ø. A zero that has a slash through it is called a "Swedish zero". Look at *your* computer's keyboard, and see whether its zero is Swedish.

The SHIFT key. One of the keys looks like this: [$ 4]. If you press it, you'll type a 4. Here's how to type a dollar sign instead: hold down the SHIFT key; and while you keep holding down the SHIFT key, tap the [$ 4] key. Here's the general rule: if a key has two symbols on it, and you want to type the top symbol, hold down the SHIFT key.

About the SHIFT key, beginners often make two boo-boos. The first boo-boo is to forget to press it. For example, suppose you want to type a dollar sign, and you press the [$ 4] key, but you forget to hold down the SHIFT key. Then you'll be typing a 4 instead.

The other boo-boo is to try to hit the SHIFT key and another key at exactly the same time. You can't do it; it's impossible; you'll wind up hitting one key before the other. The trick is to hold down the SHIFT key *first*; and while you keep holding it down, give the other key a light tap.

Beginners often make those boo-boos. When they realize their mistakes, they say "Oh, shit!" That's why programmers call it the "shit" key.

On most keyboards, the letters are automatically capitalized; so to type a capital R, just press the R key.

The BACKSPACE key. If you make a typing mistake, you can erase it, by pressing the BACKSPACE key, which makes the computer erase the last character you typed. On most keyboards, the BACKSPACE key has a left-arrow on it, like this: [←]. To erase *two* characters, press that key *twice*.

The RETURN key. The most important key is the RETURN key. Press the RETURN key at the end of every line you type. The RETURN key makes the computer read what you typed. The computer ignores what you type, until you press the RETURN key. If you forget to press the RETURN key at the end of the line, the computer doesn't read what you typed, and so the computer doesn't do anything, and you wonder why the computer seems broken.

GET STARTED

Make sure the various parts of your computer system are correctly attached to each other. Then turn on the power. If your computer is modern (and uses Modern Microsoft BASIC), it will say:

```
OK
```

If your computer is old-fashioned, it won't say OK. Instead, it will say some other word (such as READY). For details about *your* computer, read the appendix.

For further help, read the beginner's manual that came with your computer, and get help from the genie who handed you the computer (your salesperson or parent or boss or teacher or friend).

BE BOLD

In science fiction, computers blow up. In real life, they never do. No matter what keys you press, no matter what commands you type, you won't hurt the computer. The computer is invincible! So go ahead and experiment. If it doesn't like what you type, it will gripe at you, but so what?

When you try using the computer, you'll have trouble. It might be because you're making a mistake. It might be because the computer is broken. Or it might be because your computer is weird and works differently from the majority of computers discussed in this book. (Each computer has its own "personality", its own quirks.)

Whenever you have trouble, laugh about it, and say, "Oh, boy! Here we go again!" (If you're Jewish, you can say all that more briefly, in one word: "Oy!") Then get some help.

If you're in a computer center (or a computerized office or school or home), and other people are nearby, ask them for help. They'll gladly answer your questions, because they like to show off, and because the way *they* got to know the answers was by asking. Computer folks like to explain computers, just as priests like to explain religion. Remember: you're joining a cult! Even if you don't truly believe in "the power and glory of computers", at least you'll have a few moments of weird fun. So play along with the weird computer people, boost their egos, and they'll help you get through your initiation rite. Above all, assert yourself, and *ask questions*. "Shy guys finish last."

When dealing with the computer and the people who surround it, be friendly but also aggressive. To make sure you get your money's worth from a computer course, ask your teacher and coworkers questions, questions, questions! If you're using a computer that you yourself own, get help from the person who gave it to you.

Your town probably has a *computer club*. (To find out, ask the local schools and computer stores.) Join the club, and tell the club you'd like help with your computer. Probably some computer hobbyist will help you.

And don't forget—you can always call *me*, anytime you want free help.

CONQUER MATH

ARITHMETIC

Let's try to make the computer print the answer to 5+2. (And let's hope the computer says 7.) Type this:

```
PRINT 5+2
```

Type that carefully! Here's how. First, type the word PRINT, by pressing the P key, then the R key, then the I key, then the N key, then the T key. (You do *not* have to press the SHIFT key.) After typing the word PRINT, type a blank space (by pressing the SPACE bar). Then press the 5 key. Then type a plus sign, but be careful: on most keyboards, the plus sign is on the *top* part of a key; so to type the plus sign, you must hold down the SHIFT key. Then type the 2. When you've finished typing that line, press the RETURN key, which makes the computer read what you've typed.

Then the computer will print the answer:

```
 7
```

If your computer prints a wrong answer instead or says ERROR, you made a typing mistake. Find your mistake (by looking at the screen), and then try again to type

```
PRINT 5+2
```

Underneath the answer, the computer will print the word OK (or some old-fashioned word, such as READY). That means it's okay for you to feed the computer the next problem; the computer is ready for another.

If you want to subtract 3 from 7, type this:

```
PRINT 7-3
```

(To type the minus sign, you do *not* press the SHIFT key.) When you've finished typing that line, remember to press the RETURN key, which makes the computer read the line and print the answer. The computer will print:

```
 4
```

You can use decimal points and negative numbers. For example, if you type this—

```
PRINT -26.3+1
```

the computer will print:

```
-25.3
```

To multiply, use an asterisk. To multiply 2 by 6, type this:

```
PRINT 2*6
```

The computer will print:

```
 12
```

To divide, use a slash. So to divide 8 by 4, type this:

```
PRINT 8/4
```

The computer will print:

```
 2
```

Do *not* put commas in large numbers. To write four million, do *not* write 4,000,000; instead, write 4000000.

QUESTION MARK

If you're too lazy to type the word PRINT, type a question mark instead. For example, instead of typing—

```
PRINT 5+2
```

you can type:

```
? 5+2
```

The question mark is an abbreviation: it stands for the word PRINT.

If you prefer, you can think of the question mark as standing for the word "What's". For example, if you want to ask the computer "What's 5+2", you type:

```
? 5+2
```

E NOTATION

If the computer types a number with an E, the E means: move the decimal point.

For example, suppose the computer says the answer to a problem is:

```
8.51673E+12
```

The E means, "move the decimal point". The plus sign means, "towards the right". Altogether, the E+12 means, "move the decimal point towards the right, 12 places. So look at 8.51673, and move the decimal point towards the right, 12 places; you get 8516730000000. So when the computer says the answer is 8.51673E+12, the computer really means the answer is 8516730000000, approximately. (The exact answer might be 8516730000000.2, or 8516730000001.79, or some similar number; what the computer tells you is just an approximation.)

Suppose your computer says the answer to a problem is:

```
9.26E-04
```

After the E, the minus sign means, "towards the *left*". So look at 9.26, and move the decimal point towards the left, 4 places; you get:

.000926

So when the computer says the answer is 9.26E-04, the computer really means the answer is:

.000926

You won't have to worry much about E notation: the computer uses it only if an answer is larger than 999999 or tinier than .01. But when the computer *does* use E notation, remember to move the decimal point!

RANGE

The highest number the computer can handle is about 1E38, which is 1 followed by 38 zeros, like this:

10000000000000000000000000000000000000

If you try to go much higher, the computer will say:

```
OVERFLOW ERROR
```

The tiniest decimal the computer can handle is about 1E-38, which is a decimal point followed by 38 digits, 37 of which are zeros, like this:

.00000000000000000000000000000000000001

If you try to go tinier, the computer will have an "underflow error" and will "fake" the answer: it will give zero instead of the correct answer.

ORDER OF OPERATIONS

What does "2 plus 3 times 4" mean? The answer depends on who you ask.

To a businessman, it means "start with 2 plus 3, then multiply by 4"; that makes 5 times 4, which is 20. But to a scientist, "2 plus 3 times 4" means something different: it means "2 plus three fours", which is 2+4+4+4, which is 14.

Since computers were invented by scientists, computers think like scientists. If you type—

```
PRINT 2+3*4
```

the computer will think you mean "2 plus three fours", so it will do 2+4+4+4 and print this answer:

```
 14
```

The computer will *not* print the businessman's answer, which is 20. So if you're a businessman, tough luck!

Scientists and computers follow this rule: do multiplication and division before addition and subtraction. So if you type—

```
PRINT 2+3*4
```

the computer begins by hunting for multiplication and division. When it finds the multiplication sign between the 3 and the 4, it multiplies 3 by 4 and gets 12, like this:

```
PRINT 2+3*4
        12
```

So the problem becomes 2+12, which is 14, which the computer prints.

For another example, suppose you type:

```
PRINT 10-2*3+72/9*5
```

The computer begins by doing all the multiplications and divisions. So it does 2*3 (which is 6) and does 72/9*5 (which is 8*5, which is 40), like this:

```
PRINT 10-2*3+72/9*5
          6   40
```

So the problem becomes 10-6+40, which is 44, which is the answer the computer prints.

PARENTHESES

You can use parentheses the same way as in algebra. For example, 5-(1+1) means 5-2, which is 3. You can put parentheses inside parentheses: 10-(5-(1+1)) means 10-(5-2), which is 10-3, which is 7.

LOVELY STRINGS

STRINGS

Let's make the computer fall in love. Let's make it say, "I love you."

Type this:

```
PRINT "I LOVE YOU"
```

Type it carefully. Begin by typing the word PRINT. (If you're lazy, type a question mark instead.) Then press the SPACE bar. Then type a quotation mark, but be careful: on most keyboards, the quotation mark is on the *top* part of a key; so to type the quotation mark, you must hold down the SHIFT key. Then type I LOVE YOU. Then type another quotation mark.

At the end of the line, press the RETURN key. The computer will obey your command; it will print:

```
I LOVE YOU
```

You can change the computer's personality. For example, if you give this command—

```
PRINT "I HATE YOU"
```

the computer will reply:

```
I HATE YOU
```

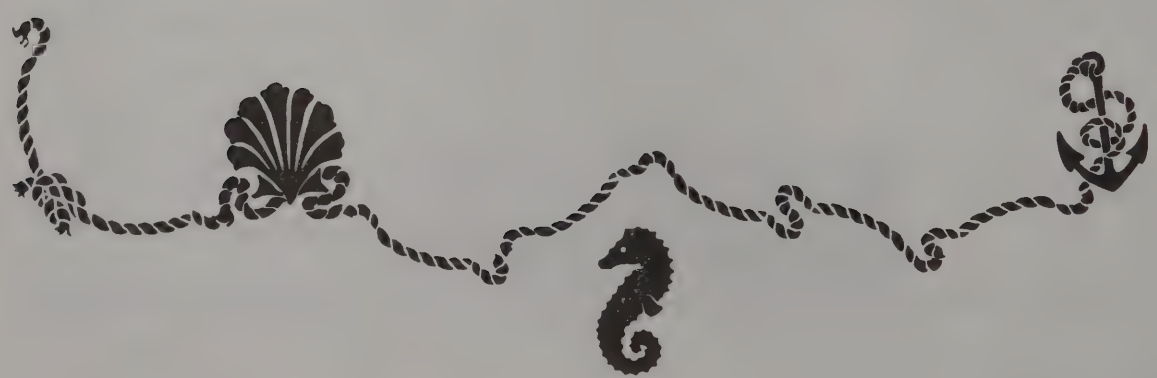
Notice that to make the computer print a message, you must put the message between quotation marks. The quotation marks make the computer copy the message without worrying about what the message means. For example, if you misspell "I love you", and type—

```
PRINT "AIEEE LUF YA"
```

the computer will still copy the message (without worrying about what it means); the computer will print:

```
AIEEE LUF YA
```

A *string* is any collection of characters, such as "I LOVE YOU" or "AIEEE LUF YA" or "76 TROMBONES" or "GO AWAY!!!" or "XYPW EXR///746". The computer will print whatever string you wish; but remember to put the string in quotation marks.



STRINGS VERSUS NUMBERS

The computer can handle two types of expressions: *strings* and *numbers*. Put strings (such as "I LOVE YOU") in quotation marks. Numbers (such as 2+2) do *not* go in quotation marks.

What happens if you make a boo-boo? What happens if you accidentally put a number in quotation marks? For example, what happens if you accidentally put 2+2 in quotation marks, like this:

```
PRINT "2+2"
```

The quotation marks make the computer think "2+2" is a string instead of a number. Since the computer thinks "2+2" is a string, it copies the string, without analyzing what it means; the computer will print:

```
2+2
```

It will *not* print 4.

What happens if you make the opposite kind of boo-boo? What happens if you mean to type a string, but forget the quotation marks? For example, what happens if you mean to type "I LOVE YOU" but forget the quotation marks, and accidentally type this instead:

```
PRINT I LOVE YOU
```

Because you forgot the quotation marks, the computer thinks the I LOVE YOU is a weird number instead of a string; but the computer doesn't know what kind of weird number it is, since the computer doesn't know the meaning of love. Whenever the computer is confused, it either gripes at you or prints a zero. In this particular example, it will print a zero, like this:

```
 0
```

So if you incorrectly tell the computer to say it loves you, it will say zero.

LONG COMMANDS ON SMALL SCREENS

Try typing this command:

```
PRINT "I LIKE TO EAT AND DRINK AND GAZE INTO YOUR EYES AS WE DREAM ABOUT LOVE"
```

That command's long. It won't fit on a single line of your screen, if your screen's small; but go ahead and type the command anyway. When you reach the screen's right edge, the computer automatically moves you to the line below, so you can continue typing. Do *not* press the RETURN key, until you've typed the entire command.

After typing the entire command, press the RETURN key. The computer will print:

```
I LIKE TO EAT AND DRINK AND GAZE INTO YOUR EYES AS WE DREAM ABOUT LOVE
```

EASY PROGRAMS

PROGRAMMING

Now that you've had fun experimenting with the word PRINT, you're ready for the big-time; you're ready to program the computer! Programming the computer consists of three steps. . . .

Step 1: type the word NEW. That tells the computer you're going to invent a new program. After you type the word NEW, press the RETURN key. The computer will say OK. (If your computer is old-fashioned, it will say some other word instead, such as READY.)

Step 2: type your program. A *program* is a list of numbered commands. For example, suppose you want the computer to say:

```
I LOVE YOU
YOU TURN ME ON
LET'S GET MARRIED
```

Type this program:

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
3 PRINT "LET'S GET MARRIED"
```

Every line of the program must be numbered; you must type those numbers. Every line of the program must say PRINT; you must type the word PRINT. (Or, if you're too lazy to type the word PRINT, you can type a question mark instead.) Every line of that program must have quotation marks; you must type the quotation marks. At the end of each line, press the RETURN key.

Step 3: type the word RUN. That tells the computer to look at the program you've written, and run through it. After you type the word RUN, press the RETURN key. The computer will print everything you requested; it will print:

```
I LOVE YOU
YOU TURN ME ON
LET'S GET MARRIED
```

Then the computer will say OK.

Let's review that procedure. You type:

```
NEW
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
3 PRINT "LET'S GET MARRIED"
RUN
```

The numbered instructions (1, 2, and 3) are called the *program*. Above the program, you type the word NEW; below the program, you type the word RUN. The words NEW and RUN are *not* numbered. After you've done all that correctly (and pressed the RETURN key after the word RUN), the computer will obey the program and print what you requested.

Each line of the program must be numbered. Putting a number at the beginning of a line makes the line become part of your program; the computer won't obey the line until the program is RUN. Here's the rule: a number in front of the line tells the computer to wait until you type the word RUN.

For another example, try typing this:

```
NEW
1 PRINT "I LONG"
2 PRINT 2+2
3 PRINT "U"
RUN
```

At the end of each line, remember to press the RETURN key. When you press the RETURN key after the word RUN, the computer obeys the program. The first line makes the computer print "I LONG"; the second line makes the computer add 2 and 2, so the computer prints the number 4; the third line makes the computer print the letter U. Altogether, the computer prints:

```
I LONG
 4
U
```

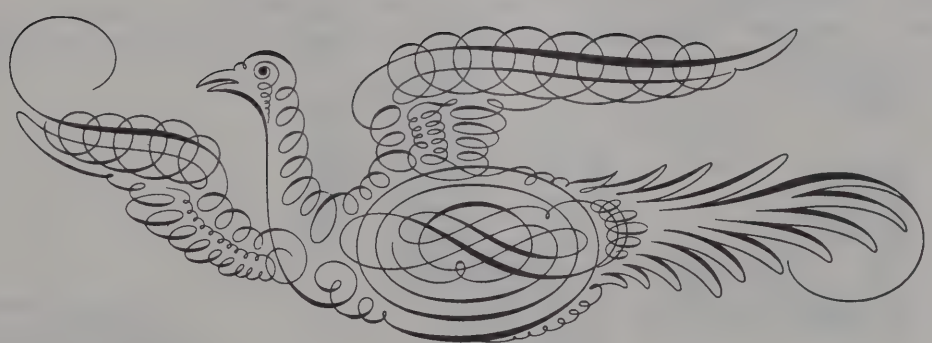
Yes, the computer says it longs for you!

In that example, the computer indents the 4, so that the 4 has a blank space in front of it. That's because of this rule: the computer automatically puts a blank space in front of every positive number.

After trying those examples, you can feed the computer another program, by going through the three steps again: type the word NEW, then type the program, then type the word RUN. (Remember to press the RETURN key at the end of each line, and remember that the commands NEW and RUN are *not* numbered.)

Those three steps (NEW then program then RUN) are all you have to know about programming! C'mon, write some programs! It's easy! Try it; you'll have lots of fun!

A person who writes a program is called a *programmer*. Congratulations: *you're* a programmer!



MULTIPLE COPIES

Suppose you want to make multiple copies of this poem:

```
SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
```

First, turn the poem into a program, by typing:

```
NEW
1 PRINT "SPRING HAS SPRUNG."
2 PRINT "THE GRASS HAS RIZ."
3 PRINT "I WONDER WHERE"
4 PRINT "THE FLOWERS IS."
```

Then type the word RUN (and press the RETURN key). The computer will print the poem. If you type the word RUN again, the computer will print the poem again. Each time you type the word RUN, the computer will print the poem. So to print many copies of the poem, just type the word RUN many times, like this:

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
READY

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
READY

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
etc.

If you hate poetry and all other forms of art, cheer up: by using the same technique, you can make the computer type many copies of your favorite flubbed-up business letter.

HOW TO PRINT ON PAPER

Now you can make the computer print messages, poems, and business letters on your screen. But if you want to mail those messages and letters to your friends, you have a problem: the post office refuses to mail your screen! And it's hard to stuff your screen into an envelope! So to mail the computer's wise words to your friends, you must tell the computer to print on paper instead of on the screen.

To make the computer print on paper, attach the computer to a *printer* (which looks like a typewriter and handles paper). On the computer's keyboard, type a special command that makes the computer use the printer. For details, read the appendix about your computer.

ERRORS

HOW TO CORRECT A MISTAKE

If you make a mistake, you can correct it in three ways. . . .

If you notice the mistake before you press the RETURN key, correct the mistake by using the BACKSPACE key.

If you mess up the whole program, rewrite it, by typing the word NEW and then starting from the beginning again.

If you mess up a line, retype it underneath. For example, suppose you type—

```
1 PRIMT "I LOVE YOU"
```

and then you notice that you misspelled the word PRINT. Just retype the line, so your screen looks like this:

```
1 PRIMT "I LOVE YOU"
1 PRINT "I LOVE YOU"
```

Here's another example. . . . Suppose you type—

```
1 PRINT "I LIKE COFFEE"
2 PRINT "I LIKE TEE"
3 PRINT "I LIKE THE GIRLS"
4 PRINT "AND THE GIRLS LIKE ME"
```

and then you notice you misspelled the word TEA in line 2. Underneath the program, retype that line, so your screen looks like this:

```
1 PRINT "I LIKE COFFEE"
2 PRINT "I LIKE TEE"
3 PRINT "I LIKE THE GIRLS"
4 PRINT "AND THE GIRLS LIKE ME"
2 PRINT "I LIKE TEE"
```

GRIPES

If the computer gripes, don't worry: just correct your mistake. For example, suppose you type—

```
1 PRIMT "ROSES ARE RED"
2 PRINT "CABBAGE IS GREEN"
3 PRINT "MY FACE IS FUNNY"
4 PRINT "BUT YOURS IS A SCREAM"
```

and you don't notice that you misspelled PRINT in line 1. If you type RUN, the computer will try to run the program but will be confused by line 1, so it will gripe at you. It will say:

```
SYNTAX ERROR IN 1
```

That means the computer hasn't the faintest idea of what you're talking about in line 1. Your next step is to correct the error. To do that, just retype line 1 correctly, and then type the word RUN again. So altogether, the conversation looks like this:

You type this, but it contains an error→
```
1 PRIMT "ROSES ARE RED"
2 PRINT "CABBAGE IS GREEN"
3 PRINT "MY FACE IS FUNNY"
4 PRINT "BUT YOURS IS A SCREAM"
RUN
```

The computer gripes at you→
```
SYNTAX ERROR IN 1
```

You type this, to correct the error→
```
1 PRINT "ROSES ARE RED"
RUN
```

The computer recites the poem→
```
ROSES ARE RED
CABBAGE IS GREEN
MY FACE IS FUNNY
BUT YOURS IS A SCREAM
```

COMMON BLOOPERS

If you're like most beginners, you'll make these mistakes soon—if you haven't made them already!

You type the letter O instead of zero, or type zero instead of the letter O. Your typing looks correct to you, but the computer gripes.

You type a command (such as RUN) but you forget to press the RETURN key afterwards. The computer keeps waiting for you to press it. Since you don't realize the computer's waiting for you, and since the computer isn't replying, you think the computer is broken.

You tell the computer to PRINT a message, but you forget to put the message in quotation marks. So the computer gripes or prints a zero instead.

You start typing a new program, but forget to type the word NEW. So when you type RUN, the computer runs a mish-mash of the new program with the previous program, and reprints some messages from the previous program.

REARRANGING YOUR PROGRAM

You don't have to type your program in order. For example, suppose you type:

```
2 PRINT "DRINKS"
1 PRINT "MOMMY"
4 PRINT "BLOOD"
3 PRINT "MY"
```

In its mind, the computer automatically rearranges the program, so the numbers are in increasing order, like this:

```
1 PRINT "MOMMY"
2 PRINT "DRINKS"
3 PRINT "MY"
4 PRINT "BLOOD"
```

To find out what's in the computer's mind, type the word LIST, like this:

You type this program→→→

If you type LIST, the computer will show you the result of all that reasoning; the computer will type:

```
10 PRINT "KISS"
20 PRINT "MY"
32 PRINT "PRETTY"
50 PRINT "FEET"
90 PRINT "AND CRY"
```

If you type RUN, the computer will print:

```
KISS
MY
PRETTY
FEET
AND CRY
```

If you think about that example, you'll notice two things:

To revise a line, just retype it.

To erase line 80, just type—
80
with nothing else on the line.

The word NEW erases your program from the computer's mind, to make way for a new one. But the word RUN does *not* erase the program; after the run, you can continue inserting, deleting, and revising lines.

If your first line is numbered 1, and your second line is numbered 2, you can't squeeze a line between them, since decimals aren't allowed. So expert programmers number their lines 10, 20, 30, 40, . . . instead of 1, 2, 3, . . . That way, if your first line is numbered 10 and your second line is numbered 20 and you want to insert an extra line between them, you can call it 15; and if you later want to insert an extra line between 10 and 15, you can call it 12; and if you want to insert another line between 10 and 12, you can call it 11.

If your program's too long to fit on your screen, tell the computer to list *part* of your program. For example, to list just lines 30 through 80, type this:

```
LIST 30-80
```

To list from line 30 to the end of your program, type this:

```
LIST 30-
```

To list from the beginning of your program up to line 80, type this:

```
LIST -80
```

To list just line 30, type this:

```
LIST 30
```

To delete lines 30 through 80, type this:

```
DELETE 30-80
```

Some computers handle the words LIST and DELETE slightly differently. To find out whether your computer's different, check the appendix.

TRICKY PRINTING

SEMICOLONS

Run this program:

```
10 PRINT "FAT";
20 PRINT "HER"
```

Line 10 makes the computer print FAT; and line 10 ends with a semicolon. The semicolon makes the computer print the next item on the same line; so the computer will print HER on the same line, like this:

```
FATHER
```

This command shows what happens to an evil king on a boat:

```
PRINT "SIN";"KING"
```

The computer will print SIN, and will print KING on the same line, like this:

```
SINKING
```

SPACES

Sometimes life consists of joy; sometimes life consists of sorrow; but sometimes life consists merely of a numb, empty feeling that's neither joy nor sorrow. To express that feeling, run this program:

```
10 PRINT "JOY"
20 PRINT
30 PRINT "SORROW"
```

Line 10 makes the computer print JOY. Line 20 makes the computer print a blank, empty line, underneath JOY. Altogether, the computer will print:

```
JOY

SORROW
```

Suppose you want the computer to print this letter:

```
DEAR JOAN,
  THANK YOU FOR THE BEAUTIFUL
NECKTIE.  JUST ONE PROBLEM--
I HATE NECKTIES!
              LOVE,
              FRED-THE-HIPPIE
```

This program prints it:

```
10 PRINT "DEAR JOAN,"
20 PRINT "  THANK YOU FOR THE BEAUTIFUL"
30 PRINT "NECKTIE.  JUST ONE PROBLEM--"
40 PRINT "I HATE NECKTIES!"
50 PRINT "              LOVE,"
60 PRINT "              FRED-THE-HIPPIE"
```

In the program, each line contains two quotation marks. To make the computer indent a line, put blank spaces *after* the first quotation mark.

Try typing this command:

```
PRINT -3;"IS MY FAVORITE NUMBER"
```

Whenever the computer prints a number, it prints a blank space afterwards; so the computer will print a blank space after the -3, like this:

```
-3 IS MY FAVORITE NUMBER
```

space

This command tells what to put in your coffee:

```
PRINT 7;"DO";"NUTS"
```

The computer prints 7 and DO and NUTS. Since 7 is a number, the computer prints a blank space after the 7. The computer prints another blank space *before* every number that's positive; so the computer prints another blank space before the 7, like this:

```
 7 DONUTS
```

spaces

Hey, man, if you're feeling cool, like maybe this command expresses your feelings:

```
PRINT "THE TEMPERATURE IS";4+25;"DEGREES"
```

The computer prints THE TEMPERATURE IS, then 4+25 (which is 29), then DEGREES. Since 29 is a positive number, the computer prints a blank space before and after the 29:

```
THE TEMPERATURE IS 29 DEGREES
```

spaces

Use this command if you're even colder:

```
PRINT "THE TEMPERATURE IS";4-25;"DEGREES"
```

The computer prints THE TEMPERATURE IS, then 4-25 (which is -21), then DEGREES. Since -21 is a number, the computer prints a space after it; but since -21 is *not* positive, the computer does *not* print a space before it. The computer prints:

```
THE TEMPERATURE IS-21 DEGREES
```

no space space

That looks ugly; it would look prettier if there were a space before the -21. To insert a space, put the space inside quotation marks:

```
PRINT "THE TEMPERATURE IS ";4-25;"DEGREES"
```

inserted space, before the quotation mark

Then the computer will print:

```
THE TEMPERATURE IS -21 DEGREES
```

inserted space

By using semicolons, you can make the computer do many numerical calculations at once. For example, suppose you want the computer to do 6+2, and also 6-2, and also 6*2, and also 6/2. This command makes the computer do all those calculations at once:

```
PRINT 6+2;6-2;6*2;6/2
```

The computer prints the four answers:

```
 8  4  12  3
```

The computer prints spaces between the answers, because the computer prints a space after every number (and an additional space before every number that's positive).

CLEARING THE SCREEN

If your computer's keyboard is fancy, it includes a CLEAR key. If you press the CLEAR key, the computer erases the entire screen, so that the screen becomes cleared and empty.

Though pressing the CLEAR key erases the screen, it does *not* erase the computer's memory; the computer still remembers what program you were working on. The only way to erase the computer's memory is to type the word NEW.

So whenever you want to begin a new program, type the word NEW, which erases the previous program; pressing the CLEAR key is *not* sufficient.

You can begin your program with this line:

```
10 CLS
```

It makes the computer automatically CLear the Screen, before doing the rest of the program.

For example, run this program:

```
10 CLS
20 PRINT "I'M SITTING ON TOP OF THE WORLD"
```

Line 10 makes the computer CLear the Screen. Since the screen's been cleared, line 20 prints this message *at the top of the screen*:

```
I'M SITTING ON TOP OF THE WORLD
```

On some computers, the CLEAR key is missing and the CLS command doesn't work: to find out about *your* computer, see the appendix.

GO ANYWHERE

GO TO

This program makes the computer go wild:

```
10 PRINT "CAT"
20 PRINT "DOG"
30 GO TO 10
```

Line 10 makes the computer print CAT. Line 20 makes it print DOG. Line 30 makes it go back to line 10 again, so it prints CAT again. Then it prints DOG again, and comes to line 30 again, which makes it go back to line 10 again, print CAT and DOG again, and then jump back again. . . . The computer will print the words CAT and DOG again and again, like this:

```
CAT
DOG
CAT
DOG
CAT
DOG
CAT
DOG
CAT
DOG
etc.
```

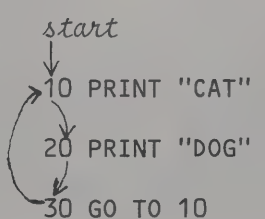
The computer will try to print the words CAT and DOG again and again, *forever*. Try running that program; you'll have fun watching the computer go crazy. The computer will print the words CAT and DOG on every line of your screen; and even when all the lines of your screen have been used, the computer still doesn't stop: all the CATs and DOGs rise to the top of the screen and finally off the screen, to make way for new CATs and DOGs to appear at the bottom of the screen. You'll see a blur, as thousands of CATs and DOGs go flying up the screen.

If you told the computer to use the printer (which prints on paper), you'll see reams of paper spewing out, saying CAT and DOG; soon the whole floor will be buried under piles of paper that say CAT and DOG thousands of times!

To stop this madness, you must give the computer some kind of "jolt" that will put it out of its misery. And now we come to the controversial part of this book. To put the computer out of its misery, you must give it an *abortion*. To give the abortion, press the BREAK key. That makes the computer stop its labor pains; it will *abort your program* and say OK. Then you can give any command, such as LIST.

(On some computers, the BREAK key is missing or acts strangely. To find out about *your* computer, check the appendix.)

Here's a picture of that program:



The computer follows the arrows, which make it go round and round in a *loop*. Since the computer will try to go round and round the loop forever, the loop is called *infinite*. The only way to stop an infinite loop is to give an abortion.

In that program, you typed just three lines; but since the third line said to GO TO the beginning, the computer does an infinite loop. By saying GO TO, you can make the computer do an infinite amount of work. Moral: *the computer can turn a finite amount of human energy into an infinite amount of good.* Putting it another way: *the computer can multiply your abilities by infinity.*

LOVE

Let's spread love throughout the world! Type this program; it makes the computer multiply your love:

```
10 PRINT "LOVE"
20 GO TO 10
```

Even though you typed the word LOVE just once, the computer will print it many times, like this:

```
LOVE
LOVE
LOVE
LOVE
LOVE
etc.
```

The computer will print repeatedly, until you abort its love.

For more lovely fun, put a semicolon after LOVE, like this:

```
10 PRINT "LOVE";
20 GO TO 10
```

The semicolon makes the computer print LOVE *next to* LOVE:

```
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
etc.
```

If you put a space after LOVE, like this—

```
10 PRINT "LOVE ";
20 GO TO 10
```

the computer will print a space after each LOVE:

```
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
etc.
```

For an even wilder love orgy, put *nine* spaces after LOVE:

```
10 PRINT "LOVE         ";
20 GO TO 10
```

The computer will print:

```
LOVE         LOVE         LOVE         LOVE         LOVE         LOVE         LO
VE         LOVE         LOVE         LOVE         LOVE         LOVE         LOVE
         LOVE         LOVE         LOVE         LOVE         LOVE         LOVE
       LOVE         LOVE         LOVE         LOVE         LOVE         LOVE
     LOVE         LOVE         LOVE         LOVE         LOVE         LOVE
etc.
```

Instead of putting *nine* spaces after LOVE, try a different number of spaces, and see what happens. Experiment! You'll get strange optical illusions, where the word LOVE seems to march across the screen, or blink, or even hop!

Instead of making the computer print LOVE, try making it print an advertisement (such as BUY JOE'S MEAT), or a political slogan (such as STOP POLLUTION), or your name, or something else that makes you feel emotional. Like a dog, the computer imitates its master's personality. If your computer acts "cold an heartless", it's because *you* are.

SMELLY POETRY

Suppose you want to send this poem to all your friends:

```
I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!
```

Just run this program:

```
10 PRINT "I'M HAVING TROUBLE"
20 PRINT "WITH MY NOSE."
30 PRINT "THE ONLY THING IT DOES IS:"
40 PRINT "BLOWS!"
50 PRINT
60 GO TO 10
```

Lines 10-40 print the poem. Line 50 prints a blank line at the end of the poem. Line 60 makes the computer do all that repeatedly, so the computer will print:

```
I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!

I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!

I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!
```

etc.

The computer will print infinitely many copies of the poem—unless you give it an abortion. If you activate the computer's printer, the copies will be printed on paper, so you can mail them to your nosy friends.

LIFE IS AN INFINITE LOOP

A program that makes the computer do the same thing again and again forever is an infinite loop. Some humans act just like computers. Those humans do the same thing again and again. Every morning they GO TO work, and every evening the GO TO home. GO TO work, GO TO home, GO TO work, GO TO home, GO TO work, GO TO home, . . . their lives are sheer drudgery. They're caught in an infinite loop. Their only escape is for some nice guy to come along and jerk them out of their misery—by giving them an abortion.

(Somewhere in that paragraph lurks a sinister comment about our society. Is it that "abortion is good"? Is it that "everybody needs to get jerked off"? Is it that "when you tell a human where to go, you're asking for trouble"? Is it that "our whole society is just a bunch of drunks who spend their lives getting looped"? Or is it that "the ultimate infinite trip is a bummer"? My analyst hopes to help me work this out.)

Go into your bathroom, get your bottle of shampoo, and look at the instructions on the back. A typical bottle has three instructions:

Lather.
Rinse.
Repeat.

Those instructions say to lather, then rinse, then repeat—which means to lather again, then rinse again, then repeat again—which means to lather again, then rinse again, then repeat again—which means to lather again, then rinse again, then repeat again. . . . If you follow those instructions, you'll never finish washing your hair! The instructions are an infinite loop! The instructions are a program: they program you to use lots of shampoo! That's how infinite loops help sell shampoo.

COMPUTER TORTURE

This program tortures the computer:

```
10 GO TO 10
```

Line 10 tells the computer to go to line 10, which tells the computer to go to line 10, which tells the computer to go to line 10. . . . The computer will do line 10 again and again, forever. So the computer will spend the rest of its life just mumbling to itself. That program has turned your beautiful computer into a bumbling, mumbling idiot. If you're mean enough to give your computer that program, please give it an abortion! Otherwise, a member of the Association for Prevention of Cruelty to Computers will shoot you.

To make the program even meaner, insert line 5, like this:

```
5 CLS
10 GO TO 10
```

Line 5 makes the computer CLear the Screen, so that the entire screen becomes blank. And line 10 renders the computer completely useless: the computer will ignore anything you type, since the computer is busy doing "10 GO TO 10" forever. You now have a computer whose screen is permanently blank and that ignores anything you type on the keyboard! To be really mean, walk up to your best friend's computer, type that program, type the word RUN, and watch your friend suffer when the screen and the keyboard both stop working! (And see how long your friend remains your "friend"!) The only way your (ex-)friend can fix the computer is to give an abortion or turn off the power.

(Remember: *most* computers understand CLS, but *yours* might not. To find out about *your* computer, check the appendix.)

To elaborate on that program, add line 7, like this:

```
5 CLS
7 PRINT "I REFUSE TO WORK FOR YOU"
10 GO TO 10
```

When you RUN that program, the screen goes blank, then the computer says it refuses to work for you, and then line 10 does indeed make the computer refuse to work for you! The computer will refuse all your work—it'll just mumble to itself—until you give it an abortion.

Now that I've turned you into a computer saboteur, the whole world will worry that you'll kidnap computers, hold them hostage, and threaten to turn them into "10 GO TO 10" idiots! You can cause a moral dilemma for traditional Catholics, who fear giving abortions and are even more afraid of "pulling the plug". Is God against abortions for computers?

DEAD SNAKES

Here's a poor, innocent program:

```
10 PRINT "YOUR MOTHER"
20 PRINT "HATES ANYONE WHO"
30 PRINT "KISSES DEAD SNAKES"
```

It makes the computer print:

```
YOUR MOTHER
HATES ANYONE WHO
KISSES DEAD SNAKES
```

To make that program more diabolical, insert line 15, like this:

```
          10 PRINT "YOUR MOTHER"
New line→→15 GO TO 30
          20 PRINT "HATES ANYONE WHO"
          30 PRINT "KISSES DEAD SNAKES"
```

The computer does line 10, which prints YOUR MOTHER. Then it does line 15, which makes it skip ahead to line 30, which prints KISSES DEAD SNAKES. So altogether, the computer prints:

```
YOUR MOTHER
KISSES DEAD SNAKES
```

In that program, line 15 made the computer skip ahead to line 30; in other words, it made the computer skip over line 20.

VARIABLES

NUMERIC VARIABLES

A letter can stand for a number. For example, X can stand for the number 47, as in this program:

```
10 X=47
20 PRINT X+2
```

Line 10 says X stands for the number 47. In other words, X is a name for the number 47. Line 20 says to print X+2; since X is 47, X+2 is 49; so the computer will print 49. That's the only number the computer will print; it will not print 47.

A letter that stands for a number is called a *numeric variable*. In that program, X is a numeric variable; it stands for the number 47. The *value* of X is 47. Line 10 is called an *assignment statement*, because it *assigns* 47 to X.

Another example:

```
10 Y=38
20 PRINT Y-2
```

Line 10 says Y is a numeric variable that stands for the number 38. Line 20 says to print Y-2; since Y is 38, Y-2 is 36; so the computer will print 36.

Example:

```
10 B=8
20 PRINT B*3
```

Line 10 says B is 8. Line 20 says to print B*3, which is 8*3, which is 24; so the computer will print 24.

One variable can define another:

```
10 M=6
20 P=M+1
30 PRINT M*P
```

Line 10 says M is 6. Line 20 says P is M+1, which is 6+1, which is 7; so P is 7. Line 30 says to print M*P, which is 6*7, which is 42; so the computer will print 42.

A value can change:

```
10 F=4
20 F=9
30 PRINT F*2
```

Line 10 says F's value is 4. Line 20 changes F's value to 9, so line 30 prints 18.

On the left side of the equal sign, you must have *one* variable:

Allowed	Not allowed	Not allowed
P=M+1	P-M=1	1=P-M
↑	↑ ↑	↑
one variable	*two variables*	*not a variable*

The variable on the left side of the equation is the only one that changes. For example, the statement P=M+1 changes the value of P but not M. The statement A=B changes the value of A but not B:

```
10 A=1
20 B=7
30 A=B
40 PRINT A+B
```

Line 30 changes A, to make it equal B; so A becomes 7. Since both A and B are now 7, line 40 prints 14.

"A=B" has a different effect than "B=A". That's because "A=B" changes the value of A (but not B); "B=A" changes the value of B (but not A). Compare these programs:

```
10 A=1                10 A=1
20 B=7                20 B=7
30 A=B                30 B=A
40 PRINT A+B          40 PRINT A+B
```

In the left program (which you saw before), line 30 changes A to 7, so both A and B are 7; line 40 prints 14. In the right program, line 30 changes B to 1, so both A and B are 1; line 40 prints 2.

If you don't define a numeric variable, the computer assumes it's zero:

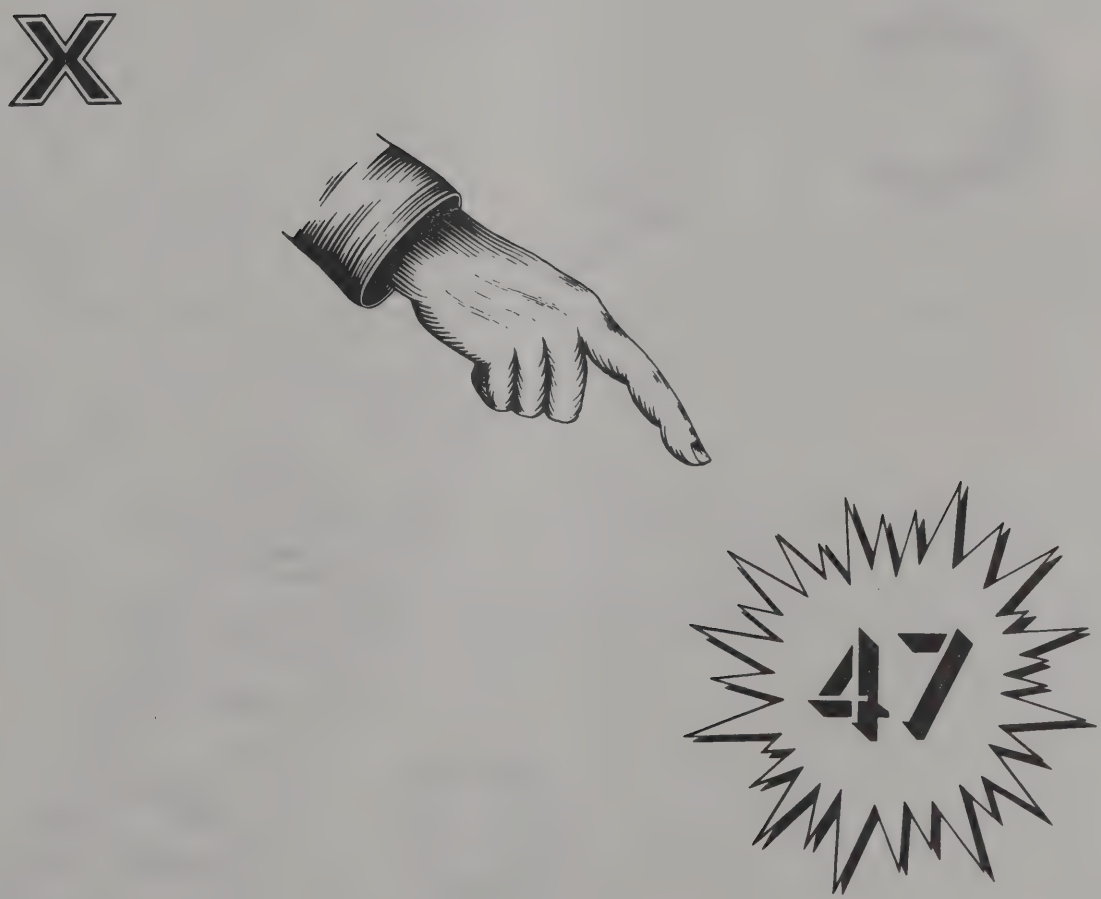
```
10 PRINT R
```

Since R hasn't been defined, line 10 prints zero.

The computer doesn't look ahead:

```
10 PRINT J
20 J=5
```

When the computer encounters line 10, it doesn't look ahead to find out what J is. As of line 10, J is still undefined, so the computer prints zero.



WHEN TO USE VARIABLES

When should you use variables? Here's a practical example. . . .

Suppose you're selling something that costs $1297.43, and you want to do these calculations:

multiply $1297.43 by 2
multiply $1297.43 by .05
add $1297.43 to $483.19
divide $1297.43 by 37
subtract $1297.43 from $8598.61
multiply $1297.43 by 28.7

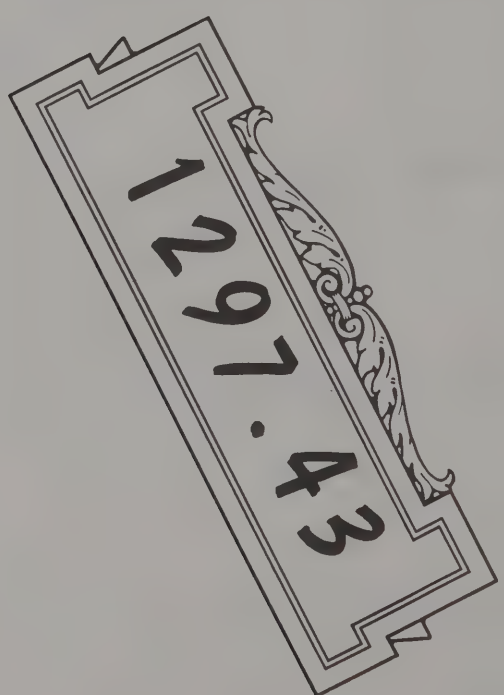
To do those six calculations, you could run this program:

```
10 PRINT 1297.43*2; 1297.43*.05; 1297.43+483.19; 1297.43/37; 8598.61-1297.43
20 PRINT 1297.43*28.7
```

But that program's silly, since it contains the number 1297.43 six times. This program's briefer, because it uses a variable:

```
10 C=1297.43
20 PRINT C*2; C*.05; C+483.19; C/37; 8598.61-C; C*28.7
```

So whenever you need to use a number several times, turn the number into a variable; it will make your program briefer.



STRING VARIABLES

A string is any collection of characters, such as "I LOVE YOU". Each string must be in quotation marks. A letter can stand for a string—if you put a dollar sign after the letter, like this:

```
10 G$="DOWN"
20 PRINT G$
```

Line 10 says G$ stands for the string "DOWN". Line 20 prints:

```
DOWN
```

In that program, G$ is a variable. Since it stands for a string, it's called a *string variable*. Every string variable must end with a dollar sign; the dollar sign is supposed to remind you of a fancy S, which stands for String. Line 10 is pronounced, "G String is DOWN".

If you're paranoid, you'll love this program:

```
10 L$="THEY'RE LAUGHING AT YOU"
20 PRINT L$
30 PRINT L$
40 PRINT L$
```

Line 10 says L$ stands for the string "THEY'RE LAUGHING AT YOU". Lines 20, 30, and 40 make the computer print:

```
THEY'RE LAUGHING AT YOU
THEY'RE LAUGHING AT YOU
THEY'RE LAUGHING AT YOU
```

The computer can recite nursery rhymes:

```
10 H$="HICKORY, DICKORY, DOCK!"
20 M$="THE MOUSE"
30 C$="THE CLOCK"
40 PRINT H$
50 PRINT M$;" RAN UP ";C$
60 PRINT C$;" STRUCK ONE"
70 PRINT M$;" RAN DOWN"
80 PRINT H$
```

Lines 10-30 define H$, M$, and C$. Lines 40-80 make the computer print:

```
HICKORY, DICKORY, DOCK!
THE MOUSE RAN UP THE CLOCK
THE CLOCK STRUCK ONE
THE MOUSE RAN DOWN
HICKORY, DICKORY, DOCK!
```

If you don't define a string variable, the computer assumes it's blank:

```
10 PRINT F$
```

Since F$ hasn't been defined, line 10 makes the computer print a line that says nothing; the line the computer prints is blank.

INPUT

STRING INPUT

Humans ask questions. So if you want to turn the computer into a human, you must make it ask questions too. To make the computer ask a question, use the word INPUT. This program makes the computer ask for your name:

```
10 INPUT "WHAT IS YOUR NAME";N$
20 PRINT "I HATE ANYONE WHOSE NAME IS ";N$
```

Line 10 makes the computer ask "WHAT IS YOUR NAME?" and then wait for you to answer; your answer will be called N$. For example, if you answer JOEY, then N$ is JOEY. Line 20 makes the computer print:

```
I HATE ANYONE WHOSE NAME IS JOEY
```

Here's the whole conversation; I've underlined the parts typed by you:

You tell the computer to run→→→→→RUN
The computer asks for your name→→WHAT IS YOUR NAME? JOEY
The computer criticizes→→→→→→→→→→I HATE ANYONE WHOSE NAME IS JOEY

Try that example. Be careful! When you type line 10, which says INPUT, make sure you type the two quotation marks and the semicolon. You don't have to type a question mark; when the computer runs your program, it will automatically put a question mark at the end of the question.

Just for fun, run that program again and pretend you're somebody else:

You tell the computer to run→→→→→RUN
The computer asks for your name→→WHAT IS YOUR NAME? GAIL
The computer criticizes→→→→→→→→→→I HATE ANYONE WHOSE NAME IS GAIL

COLLEGE ADMISSIONS

This program prints a letter, admitting you to the college of your choice:

```
10 INPUT "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO";C$
20 PRINT "CONGRATULATIONS!"
30 PRINT "YOU HAVE JUST BEEN ADMITTED TO ";C$
40 PRINT "IT FITS YOUR PERSONALITY"
50 PRINT "I HOPE YOU GO TO ";C$
60 PRINT "           RESPECTFULLY YOURS"
70 PRINT "           THE DEAN OF ADMISSIONS"
```

Line 10 makes the computer ask "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO?" and then wait for you to answer; your answer will be called C$. If you'd like to be admitted to HARVARD, you'll be pleased:

```
You tell the computer to run→→RUN
The computer asks→→→→→→→→→→→→→→WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO? HARVARD
The computer admits you→→→→→→→CONGRATULATIONS!
                              YOU HAVE JUST BEEN ADMITTED TO HARVARD
                              IT FITS YOUR PERSONALITY
                              I HOPE YOU GO TO HARVARD
                                        RESPECTFULLY YOURS
                                        THE DEAN OF ADMISSIONS
```

You can choose any college you wish:

```
RUN
WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO? HELL
CONGRATULATIONS!
YOU HAVE JUST BEEN ADMITTED TO HELL
IT FITS YOUR PERSONALITY
I HOPE YOU GO TO HELL
           RESPECTFULLY YOURS
           THE DEAN OF ADMISSIONS
```

That program consists of three parts:

1. The computer begins by asking you a question ("What college would you like to be admitted to?"). The computer's question is called the *prompt*, because it prompts you to answer.

2. Your answer (the college's name) is called *your input*, because it's information that you're *putting into* the computer.

3. The computer's reply (the admission letter) is called the *computer's output*, because it's the final answer that the computer puts out.

INPUT VERSUS PRINT

The word INPUT is the opposite of the word PRINT. The word PRINT makes the computer print information out; the word INPUT makes the computer take information in. The word PRINT makes the computer output; the word INPUT makes the computer accept your input. Input and Output are collectively called *I/O*, so the INPUT and PRINT statements are called *I/O statements*.

STORIES

Let's make the computer write a story, by filling in the blanks:

```
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED _____
                                              your name

WHO HAD A FRIEND NAMED _____
                       friend's name

_____ WANTED TO _____ _____
  your name       verb (such as "hit")   friend's name

BUT _____ DIDN'T WANT TO _____ _____
     friend's name              verb (such as "hit")  your name

WILL _____ _____ _____
      your name  verb (such as "hit")  friend's name

WILL _____ _____ _____
      friend's name  verb (such as "hit")  your name

TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE

OF _____ AND _____
     your name      friend's name
```

To write the story, the computer must ask for your name, your friend's name, and a verb. To make the computer ask, your program must say INPUT. Here's the program:

The computer asks→→→→
The computer writes→→

```
10 INPUT "WHAT'S YOUR NAME";Y$
20 INPUT "WHAT'S YOUR FRIEND'S NAME";F$
30 INPUT "IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND";V$
40 PRINT "HERE'S MY STORY...."
50 PRINT "ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED ";Y$
60 PRINT "WHO HAD A FRIEND NAMED ";F$
70 PRINT Y$;" WANTED TO ";V$;" ";F$
80 PRINT "BUT ";F$;" DIDN'T WANT TO ";V$;" ";Y$
90 PRINT "WILL ";Y$;" ";V$;" ";F$
100 PRINT "WILL ";F$;" ";V$;" ";Y$
110 PRINT "TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPIS
ODE"
120 PRINT "OF ";Y$;" AND ";F$
```

Here's a sample run:

```
WHAT'S YOUR NAME? DRACULA
WHAT'S YOUR FRIEND'S NAME? MARILYN MONROE
IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND? BITE
HERE'S MY STORY....
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED DRACULA
WHO HAD A FRIEND NAMED MARILYN MONROE
DRACULA WANTED TO BITE MARILYN MONROE
BUT MARILYN MONROE DIDN'T WANT TO BITE DRACULA
WILL DRACULA BITE MARILYN MONROE
WILL MARILYN MONROE BITE DRACULA
TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE
OF DRACULA AND MARILYN MONROE
```

Here's another run:

```
WHAT'S YOUR NAME? SUPERMAN
WHAT'S YOUR FRIEND'S NAME? KING KONG
IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND? TICKLE
HERE'S MY STORY....
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED SUPERMAN
WHO HAD A FRIEND NAMED KING KONG
SUPERMAN WANTED TO TICKLE KING KONG
BUT KING KONG DIDN'T WANT TO TICKLE SUPERMAN
WILL SUPERMAN TICKLE KING KONG
WILL KING KONG TICKLE SUPERMAN
TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE
OF SUPERMAN AND KING KONG
```

Try it: put in your own name, the name of your friend, and something you'd like to do to your friend.

This program creates a story that's fancier:

```
10 INPUT "WHAT'S YOUR NAME";Y$
20 INPUT "WHAT'S YOUR FRIEND'S NAME";F$
30 INPUT "WHAT'S THE NAME OF ANOTHER FRIEND";A$
40 INPUT "NAME A COLOR";C$
50 INPUT "NAME A PLACE";P$
60 INPUT "NAME A FOOD";D$
70 INPUT "NAME AN OBJECT";J$
80 INPUT "NAME A PART OF THE BODY";B$
90 INPUT "NAME A STYLE OF COOKING (SUCH AS BAKED OR FRIED)";S$
100 PRINT
110 PRINT "CONGRATULATIONS ";Y$
120 PRINT "YOU'VE WON THE BEAUTY CONTEST, BECAUSE OF YOUR SEXY ";B$
130 PRINT "YOUR PRIZE IS A ";C$;" ";J$
140 PRINT "PLUS A TRIP TO ";P$;" WITH YOUR FRIEND ";F$
150 PRINT "PLUS...AND THIS IS THE BEST PART OF ALL..."
160 PRINT "DINNER FOR THE TWO OF YOU AT ";A$;"'S NEW RESTAURANT"
170 PRINT "WHERE ";A$;" WILL GIVE YOU ALL THE ";S$;" ";D$;" YOU CAN EAT"
180 PRINT "CONGRATULATIONS ";Y$;"...TODAY'S YOUR LUCKY DAY..."
190 PRINT "NOW EVERYBODY WANTS TO KISS YOUR AWARD-WINNING ";B$
```

Here's a sample run:

```
WHAT'S YOUR NAME? IVAN
WHAT'S YOUR FRIEND'S NAME? FRITZ
WHAT'S THE NAME OF ANOTHER FRIEND? BRIGITTE
NAME A COLOR? RED
NAME A PLACE? TIMES SQUARE
NAME A FOOD? CHERRIES
NAME AN OBJECT? TOILET
NAME A PART OF THE BODY? ASS
NAME A STYLE OF COOKING (SUCH AS BAKED OR FRIED)? STEAMED

CONGRATULATIONS IVAN
YOU'VE WON THE BEAUTY CONTEST, BECAUSE OF YOUR SEXY ASS
YOUR PRIZE IS A RED TOILET
PLUS A TRIP TO TIMES SQUARE WITH YOUR FRIEND FRITZ
PLUS...AND THIS IS THE BEST PART OF ALL...
DINNER FOR THE TWO OF YOU AT BRIGITTE'S NEW RESTAURANT
WHERE BRIGITTE WILL GIVE YOU ALL THE STEAMED CHERRIES YOU CAN EAT
CONGRATULATIONS IVAN...TODAY'S YOUR LUCKY DAY...
NOW EVERYBODY WANTS TO KISS YOUR AWARD-WINNING ASS
```

BILLS

If you're a nasty businessman, you'll love this program:

```
10 INPUT "WHAT IS THE CUSTOMER'S FIRST NAME";F$
20 INPUT "LAST NAME";L$
30 INPUT "STREET ADDRESS";A$
40 INPUT "CITY";C$
50 INPUT "STATE";S$
60 INPUT "ZIP CODE";Z$
70 PRINT
80 PRINT F$;" ";L$
90 PRINT A$
100 PRINT C$;" ";S$;" ";Z$
110 PRINT
120 PRINT "DEAR ";F$
130 PRINT "   YOU STILL HAVEN'T PAID THE BILL."
140 PRINT "IF YOU DON'T PAY IT SOON, ";F$;","
150 PRINT "I'LL COME VISIT YOU IN ";C$
160 PRINT "AND PERSONALLY SHOOT YOU."
170 PRINT "             YOURS TRULY,"
180 PRINT "             SURE-AS-SHOOTIN'"
190 PRINT "             YOUR CRAZY CREDITOR"
```

Can you figure out what that program does?

NUMERIC INPUT

This program makes the computer predict your future:

```
10 PRINT "I WILL PREDICT WHAT WILL HAPPEN TO YOU IN THE YEAR 2000!"
20 INPUT "IN WHAT YEAR WERE YOU BORN";Y
30 PRINT "IN THE YEAR 2000, YOU WILL BECOME";2000-Y;"YEARS OLD."
```

Here's a sample run:

```
I WILL PREDICT WHAT WILL HAPPEN TO YOU IN THE YEAR 2000!
IN WHAT YEAR WERE YOU BORN? 1962
IN THE YEAR 2000, YOU WILL BECOME 38 YEARS OLD.
```

Suppose you're selling tickets to a play. Each ticket costs $2.79. (You decided $2.79 would be a nifty price, because the cast has 279 people.) This program finds the price of multiple tickets:

```
10 INPUT "HOW MANY TICKETS";T
20 PRINT "THE TOTAL PRICE IS $";T*2.79
```

This program tells you how much the "energy crisis" costs you, when you drive your car:

```
10 INPUT "HOW MANY MILES DO YOU WANT TO DRIVE";M
20 INPUT "HOW MANY PENNIES DOES A GALLON OF GAS COST";P
30 INPUT "HOW MANY MILES-PER-GALLON DOES YOUR CAR GET";R
40 PRINT "THE GAS FOR YOUR TRIP WILL COST YOU $";M*P/(R*100)
```

Here's a sample run:

```
HOW MANY MILES DO YOU WANT TO DRIVE? 300
HOW MANY PENNIES DOES A GALLON OF GAS COST? 127.9
HOW MANY MILES-PER-GALLON DOES YOUR CAR GET? 31
THE GAS FOR YOUR TRIP WILL COST YOU $ 12.3774
```

This program converts feet to inches:

```
10 INPUT "HOW MANY FEET";F
20 PRINT F;"FEET =";F*12;"INCHES"
```

Here's a sample run:

```
HOW MANY FEET? 3
 3 FEET = 36 INCHES
```

Try to convert to the metric system? This program converts inches to centimeters:

```
10 INPUT "HOW MANY INCHES";I
20 PRINT I;"INCHES =";I*2.54;"CENTIMETERS"
```

Nice day today, isn't it? This program converts the temperature from Celsius to Fahrenheit:

```
10 INPUT "HOW MANY DEGREES CELCIUS";C
20 PRINT C;"DEGREES CELCIUS =";C*1.8+32;"DEGREES FAHRENHEIT"
```

Here's a sample run:

```
HOW MANY DEGREES CELCIUS? 20
 20 DEGREES CELCIUS = 68 DEGREES FAHRENHEIT
```

See, you can write the *Guide* yourself! Just hunt through any old math or science book, find any old formula (such as F=C*1.8+32), and turn it into a program.

LOGIC

COLONS

You can put several statements on the same line, like this:

```
10 A=5: B=7: PRINT A+B
```

That line says A is 5, and B is 7, and to print A+B. So the computer will print 12.

END

The computer understands the word END:

```
10 PRINT "BUBBLE GUM"
20 END
30 PRINT "FOX"
```

The computer will print BUBBLE GUM and then end, without printing FOX. After the computer ends, it'll say OK. Then you can type any command, such as LIST (to see the program again) or RUN (to make the computer print BUBBLE GUM again) or NEW (to create a new program).

IF . . . THEN

Let's make the computer interrogate a human. Let's make the computer begin the interrogation by asking whether the human is male or female. If the human answers MALE, let's make the computer say:

```
SO IS FRANKENSTEIN
```

If the human answers FEMALE, let's make the computer say:

```
SO IS MARY POPPINS
```

If the human gives a different answer (such as SUPER-MALE or MACHO or NOT SURE or BOTH or YES), let's make the computer say—

```
PLEASE SAY MALE OR FEMALE
ARE YOU MALE OR FEMALE?
```

and force the human to answer the question correctly.

This program does it:

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN": END
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS": END
40 PRINT "PLEASE SAY MALE OR FEMALE": GO TO 10
```

Line 10 makes the computer ask "ARE YOU MALE OR FEMALE?" and wait for the human's answer, which is called A$. If the human's answer is MALE, line 20 makes the computer print SO IS FRANKENSTEIN and then end. If the human's answer is FEMALE, line 30 makes the computer print SO IS MARY POPPINS and then end. If the human's answer is neither MALE nor FEMALE, the computer skips over lines 20 and 30, so it comes to line 40, which makes it print PLEASE SAY MALE OR FEMALE and then go back to line 10, which forces the human to answer the question again.

Here's a sample run:

```
RUN
ARE YOU MALE OR FEMALE? MALE
SO IS FRANKENSTEIN
```

Here's another:

```
RUN
ARE YOU MALE OR FEMALE? FEMALE
SO IS MARY POPPINS
```

Here's another:

```
RUN
ARE YOU MALE OR FEMALE? SUPER-MALE
PLEASE SAY MALE OR FEMALE
ARE YOU MALE OR FEMALE? MALE
SO IS FRANKENSTEIN
```

Whenever you say IF, you should also say THEN. For example, line 20 (which says IF) also says THEN. Similarly, line 30 (which says IF) says THEN.

What comes between IF and THEN is called the *condition*. In line 20, the condition is:

```
A$="MALE"
```

If that condition is true (if A$ really does equal "MALE"), the computer does what comes after the word THEN, which is called the *consequence*, which is:

```
PRINT "SO IS FRANKENSTEIN": END
```

In that program, if the human makes a typing error, and answers neither MALE nor FEMALE, the computer arrives at line 40, which makes the computer gripe and tell the human to try again. So the purpose of line 40 is to react to errors. Line 40 is called an *error-handling routine* or an *error trap*. That's because an error is like a vicious animal, and the purpose of line 40 is to trap it.

Let's extend the conversation. If the human says FEMALE, let's make the computer say SO IS MARY POPPINS and then ask "DO YOU LIKE HER?" If the human says YES, let's make the computer say:

```
I LIKE HER TOO--SHE IS MY MOTHER
```

If the human says NO, let's make the computer say:

```
NEITHER DO I--SHE STILL OWES ME A DIME
```

If the human says neither YES nor NO, let's make the computer say:

```
PLEASE SAY YES OR NO
DO YOU LIKE HER?
```

Here's the program:

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN": END
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS": GO TO 100
40 PRINT "PLEASE SAY MALE OR FEMALE": GO TO 10

100 INPUT "DO YOU LIKE HER";B$
110 IF B$="YES" THEN PRINT "I LIKE HER TOO--SHE IS MY MOTHER": END
120 IF B$="NO" THEN PRINT "NEITHER DO I--SHE STILL OWES ME A DIME": END
130 PRINT "PLEASE SAY YES OR NO": GO TO 100
```

Line 30 says: if the human's answer is FEMALE, print SO IS MARY POPPINS and then go to line 100, which asks "DO YOU LIKE HER?" Lines 110 and 120 make the computer react to the human's opinion of Mary Poppins. Line 130 is like line 40: it's an error trap.

The computer is like a human; it wants to make new friends. This program makes the computer show its true feelings:

```
10 INPUT "ARE YOU MY FRIEND";A$
20 IF A$="YES" THEN PRINT "THAT'S SWELL": END
30 IF A$="NO" THEN PRINT "GO JUMP IN A LAKE": END
40 PRINT "PLEASE SAY YES OR NO": GO TO 10
```

When you type RUN, the computer asks "ARE YOU MY FRIEND?" If you say YES, the computer says THAT'S SWELL. If you say NO, the computer says GO JUMP IN A LAKE.

The most inventive programmers are kids. This program was written by a girl in the sixth grade:

```
10 INPUT "CAN I COME OVER TO YOUR HOUSE TO WATCH T.V.";A$
20 IF A$="YES" THEN PRINT "THANKS.  I'LL BE THERE AT 5 P.M.": END
30 IF A$="NO" THEN PRINT "HUMPH!  YOUR FEET SMELL, ANYWAY.": END
40 PRINT "PLEASE SAY YES OR NO": GO TO 10
```

When you type RUN, the computer asks to watch your television. If you say YES, the computer promises to come to your house at 5. If you refuse, the computer insults your feet.

Another sixth-grade girl wrote this program, to test your honesty:

```
10 PRINT "FKGJDFGKJ*#K$JSLF*/#$()$&(IKJNHBGD52:?./KSDJK$E(EF$#/JIK(*"
20 PRINT "FASDFJKL:JFRFVFJUNJI*&()JNE$#SKI#(!SERF HHW NNWAZ MAME !!!"
30 PRINT "ZBB%%%%%##)))))FESDFJK DSFE N.D.JJUJASD EHWLKD******"
40 INPUT "DO YOU UNDERSTAND WHAT I SAID";A$
50 IF A$="NO" THEN PRINT "SORRY TO HAVE BOTHERED YOU": END
60 IF A$="YES" THEN GO TO 100
70 PRINT "PLEASE SAY YES OR NO": GO TO 10

100 PRINT "SSSFJSLFKDJFL++++4567345677839XSDWFEGF/#$&**()---==!!ZZXX"
110 PRINT "###EDFHTG NVFDF MKJKF ==+--*$&% #RHFS SESD DOPEKKKNJGFD DSBS"
120 INPUT "OKAY, WHAT DID I SAY";B$
130 PRINT "YOU ARE A LIAR, A LIAR, A BIG FAT LIAR!"
```

When you type RUN, lines 10-30 print nonsense. Then the computer asks whether you understand that stuff. *If you're honest* and answer NO, the computer will apologize. But *if you pretend that you understand the nonsense*, and answer YES, the computer will print more nonsense, challenge you to translate it, wait for you to fake a translation, and then scold you for lying.

A Daddy wrote a program for his five-year-old son, John. When John types RUN, the computer asks "WHAT'S 2 AND 2?" If John answers 4, the computer says NO, 2 AND 2 IS 22. If he runs the program again and answers 22, the computer says NO, 2 AND 2 IS 4. No matter how many times he runs the program and how he answers the question, the computer says he's wrong. But when Daddy runs the program, the computer replies, YES, DADDY IS ALWAYS RIGHT. Here's how Daddy programmed the computer:

```
10 INPUT "WHAT'S YOUR NAME";N$
20 INPUT "WHAT'S 2 AND 2";A
30 IF N$="DADDY" THEN PRINT "YES, DADDY IS ALWAYS RIGHT": END
40 IF A=4 THEN PRINT "NO, 2 AND 2 IS 22": END
50 PRINT "NO, 2 AND 2 IS 4"
```

YOUR PERSONALITY

This program makes the computer act human:

```
10 PRINT "HELLO"
20 INPUT "WHAT'S YOUR NAME";N$
30 PRINT "GLAD TO MEET YOU, ";N$
40 INPUT "HOW ARE YOU FEELING TODAY";F$
50 IF F$="FINE" THEN PRINT "THAT'S GOOD": END
60 IF F$="LOUSY" THEN PRINT "TOO BAD": END
70 PRINT "I FEEL THE SAME WAY"
```

When you type RUN, the computer begins the conversation by saying HELLO and asking for your name. Then it says GLAD TO MEET YOU, followed by your name; for example, if you said your name was JOEY, the computer will say:

```
GLAD TO MEET YOU, JOEY
```

Then the computer asks how you're feeling. If you say FINE, line 50 makes the computer say THAT'S GOOD. If you say LOUSY, line 60 makes the computer say TOO BAD. If you say neither FINE nor LOUSY, the computer skips lines 50 and 60 and arrives at line 70, so it says I FEEL THE SAME WAY (which is a safe, general response).

That program makes the computer imitate an "average American". But *you're not average!* Make the computer imitate *your* personality!

The first line of your program should say—

```
10 PRINT "HELLO"
```

or—

```
10 PRINT "HI"
```

or—if you're from Texas—

```
10 PRINT "HOWDY, PARDNER!"
```

or—if you're a cool jive cat—

```
10 PRINT "HI, BABY!  WHAT'S HAPPENING?"
```

or—if you're a nurse working in a private practice—

```
10 PRINT "GOOD MORNING.  THE DOCTOR WILL BE WITH YOU SHORTLY."
```

or—if you're a punk—

```
10 PRINT "BUG OFF, OR I'LL CUT YOU UP!"
```

Underneath your opening line, you'll need an INPUT statement that makes the computer ask the human a question. Underneath the INPUT statement, you'll need several IF statements that make the computer react to the human's possible responses.

Try to make your program long, so that the conversation between the computer and human lasts *several minutes*. Your program will be a mass of PRINT, INPUT, IF, and GO TO statements, plus a few ENDs.

See how human you can make your computer! See how well you can make the computer imitate *you*!

FANCY CONDITIONS

You can make the IF clause very fancy:

IF clause	Meaning
IF A$="MALE"	If A$ is "MALE"
IF A=4	If A is 4
IF A<4	If A is less than 4
IF A>4	If A is greater than 4
IF A<=4	If A is less than or equal to 4
IF A>=4	If A is greater than or equal to 4
IF A<>4	If A is not 4
IF A$<"MALE"	If A$ is a word that comes before "MALE" in the dictionary
IF A$>"MALE"	If A$ is a word that comes after "MALE" is the dictionary

The computer understands the word OR. For example, here's how to say, "If X is either 7 or 8, print the word WONDERFUL":

```
IF X=7 OR X=8 THEN PRINT "WONDERFUL"
```

That example is composed of two conditions: the first condition is "X=7"; the second condition is "X=8". Those two conditions combine, to form "X=7 OR X=8", which is called a *compound condition*.

If you use the word OR, put it between two conditions.

Right: `IF X=7 OR X=8 THEN PRINT "WONDERFUL"` *("X=7" and "X=8" are conditions.)*
Wrong: `IF X=7 OR 8 THEN PRINT "WONDERFUL"` *("8" is not a condition.)*

The computer understands the word AND. Here's how to say, "If P is more than 5 but less than 10, print TUNA FISH":

```
IF P>5 AND P<10 THEN PRINT "TUNA FISH"
```

Here's how to say, "If S is at least 60 but less than 65, print YOU ALMOST FAILED":

```
IF S>=60 AND S<65 THEN PRINT "YOU ALMOST FAILED"
```

Here's how to say, "If N is a number from 1 to 10, print "THAT'S GOOD":

```
IF N>=1 AND N<=10 THEN PRINT "THAT'S GOOD"
```

ADVICE

REMINDERS

Whenever you're confused, type this command:

```
LIST
```

You must put quotation marks around each string, and a dollar sign after each string variable:

Right	Wrong	Wrong
A$="JERK"	A$=JERK	A="JERK"

To make the computer do your program again and again, automatically, add an extra line at the bottom of your program. That extra line should tell the computer to GO TO the top line.

Here are two reasons why the computer might print too much:

You forgot to insert the word END into your program.
You forgot to type NEW; so the computer is reprinting part of the previous program.

To say "not equal to", say "less than or greater than", like this: <>.

Here's the "proper" way to write an IF statement. . . . The IF statement should contain the word THEN. Do *not* put a comma before THEN. In an IF clause, the equal sign should come last:

Right	Wrong
<=	=<
>=	=>

BECOME AN EXPERT

Programming a computer is like driving a car: the only way to become an expert is to put your hands on that mean machine and try it yourself.

Put this book next to your terminal; every time you read a paragraph, try it: type the examples and look, look, see the computer run. Invent your own variations: try typing different numbers and strings. Write your own programs. Make the computer print your name or a poem. Make it solve problems from your other courses and the rest of your life. The computer's a fantastic toy. Play with it.

If you're a student, don't wait for your instructor to give lectures and assign homework. *Act now*. You'll learn more from dealing with the computer than from the lectures or readings. Experience counts.

Let me tell you the story of Charlie. . . .

At Wesleyan University's computer center, one of the directors was having trouble making the computer print the university's payroll. He asked me for help, but I said I didn't know either. I saw a little kid sitting at one of the terminals. "Hey, Charlie," I called to him, "we're having trouble getting the payroll out."

Little Charlie came over and typed some stuff on our terminal. "The payroll will be out in a minute," he said gleefully.

Charlie was only in seventh grade. He'd never taken a computer course, because his school didn't offer one. But he'd spent the whole summer just "hanging around" our computer, and knew it better than we.

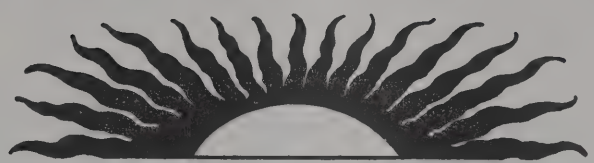
Be like Charlie. Hang around your computer. Communicate with it every day. At first, that will be even harder than talking with a cat or a tree, because the computer belongs to a different specie, a different kingdom; but keep trying. Get to know it as well as you know your best friend.

When dealing with the computer and the people who surround it, be friendly but also aggressive. To make sure you get your money's worth from a computer course, ask your teacher and coworkers questions, questions, questions!

If you're taking a French course, you might find French difficult; and if you're taking a computer course, you might find computers difficult also. But even a stupid three-year-old French kid can speak French, and even my second-graders can program the computer. They've got only one advantage over you: practice!

CHAPTER 3

HARDWARE



CHIP TECHNOLOGY

PC BOARDS

If you pry open the computer and look inside, you see a green plastic board. An electrical wiring diagram's printed on the board. But instead of being printed in ink, the diagram's printed in copper.

Since the diagram's made of copper, it conducts electricity. Therefore, it isn't just a diagram of an electrical circuit: it *is* an electrical circuit!

The green plastic board—including the circuit printed on it—is called a *printed-circuit board* (or *PC board*).

Each wire that's stamped onto the PC board is called a *trace*. If a computer company manufactures a PC board and later notices a wire on the board goes in the wrong direction, that bad wire must be cut and a new wire must be attached instead. Cutting the bad wire is called *cutting a trace*; the new wire (which is attached by an electrician instead of by a machine) is called a *jumper*. If you look inside your computer and see lots of jumpers, you know that the computer company made lots of boo-boos and then tried to correct them!

The typical computer contains *several* PC boards. In your computer, the largest and most important PC board is called the *motherboard*. The motherboard is like a prostitute: it lies on its back all day, horizontal, and its normal position is on the *bottom* of the computer's cabinet.

Besides the motherboard, the typical computer also contains several smaller PC boards. Each small PC board—which is about the size of a postcard—is called a *PC card*.

CHIPS AND DIPS

On each PC board, you'll see black bumps. If you look closely at a black bump, you'll see it has tiny little legs, so that it looks like a black caterpillar. That black caterpillar is about 1½ inches long and has about 15 pairs of legs.

Although the black caterpillar has legs, it doesn't move. It's permanently mounted on the PC board.

Each leg is made of tin and is called a *pin*.

Sadistic hobbyists play a game where they yank the caterpillars from the PC board and then throw the caterpillars across the room. That game's called "tin-pin bowling".

Hidden inside the caterpillar is a tiny metal square, called a *chip*. The chip is so tiny! It's just an eighth of an inch long, an eighth of an inch wide, and a hundredth of an inch thick!

On that tiny metal chip are etched *thousands* of microscopic electronic circuits! Because all those circuits are on the chip, the chip's also called an *integrated circuit (IC)*.

The caterpillar's only purpose is to hide and protect the chip inside it; the caterpillar's just a strange-looking package that contains the chip. Because the caterpillar's a package that has two rows of legs, it's called a *dual in-line package (DIP)*. That DIP's only purpose is to hide the chip.

Computer hobbyists are always talking about chips and DIPs. That's why, at parties, computer hobbyists serve chips and dips. And that's why computer hobbyists are called "dipchips".

If you go into a computer store and ask to buy a chip, the store will also give you the chip's DIP (the entire caterpillar). Since you've asked for a chip but also received a DIP, you might get confused and think that the caterpillar (the DIP) is the chip. But that caterpillar is *not* the chip; the chip hides inside the caterpillar.

The price for a caterpillar-and-chip depends on how fancy the chip's circuitry is. The price can be anywhere from 25¢ to $200, though $1.50 is "typical".

The chip inside the caterpillar acts as the caterpillar's brain. The caterpillar also contains a "nervous system", which consists of thin wires that run from the brain (the chip) to the legs (the pins). The wires in the caterpillar's nervous system are very thin: the diameter of each wire is about half of a thousandth of an inch.

It's fun to give each caterpillar a nickname. For example, suppose your computer contains a caterpillar named Joe and another caterpillar named Sue. When Joe wants to send electrical signals to Sue, the signals go from Joe's brain (the chip) through Joe's nervous system to Joe's legs (pins). Each of Joe's pins is attached to a trace (wire) on the PC board. The electrical signals travel from Joe's pins to the PC board's traces, which carry the signals across the PC board, until the signals reach Sue's pins. Then the signals travel through Sue's nervous system, until they reach Sue's brain (the chip).

BIPOLAR VERSUS MOS

A chip can be manufactured in two ways. The old way's called *bipolar*. The new way's called *metal-oxide semiconductor field-effect transistor* (*MOSFET*, or simply *MOS*, which is pronounced "moss").

The new way (MOS) is more popular than the old way (bipolar), because MOS costs less, consumes less electricity, and can hold more circuitry per chip. Because MOS costs so much less than bipolar, microcomputers use MOS, not bipolar.

But in spite of all its advantages, MOS has one *dis*advantage: it's slower than bipolar. A computer built from MOS chips solves problems more slowly than a computer built from bipolar chips. The most expensive maxicomputers, which brag about how fast they think, use bipolar chips to achieve the speed.

Cheaper maxicomputers and minicomputers contain a mixture of MOS chips and bipolar chips; the result is "reasonably cheap" yet "reasonably fast".

KINDS OF MOS

A MOS chip has many electrical *channels* in it. In *p-channel MOS (PMOS)*, all the channels are positively charged: they contain silicon that's been "doped" with positively charged boron. In *n-channel MOS (NMOS)*, all the channels are negatively charged: they contain silicon that's been doped by negatively charged phosphorus. In *complementary MOS (CMOS)*, the even-numbered channels are negative, and the odd-numbered channels are positive.

Which of those three kinds of MOS is best: PMOS, NMOS, or CMOS?

PMOS is too slow. Manufacturers don't use it anymore.

That leaves us with two other choices: NMOS and CMOS. Most computers use NMOS instead of CMOS, because NMOS costs less and can hold more circuitry per chip. In fact, all the popular type 2 computers (which use televisions) and type 3 computers (which use 80-column monitors) use NMOS.

But NMOS has two *dis*advantages: it's slower than CMOS, and it consumes more electricity than CMOS. NMOS consumes too much electricity run on batteries. If you want your computer to run on batteries, you must use CMOS instead.

The computer inside your digital watch and the computer inside your pocket calculator both run on batteries, because they contain CMOS chips. Type 1 computers (which use LCD displays) all use batteries and CMOS.

THE 4 PURPOSES

Inside your computer, each chip has a purpose:

If the chip's purpose is to "think", it's called a *processor chip*.

If the chip's purpose is to "remember information", it's called a *memory chip*.

If the chip's purpose is to help different parts of the computer communicate with each other, it's called an *interface chip*.

If the chip's purpose is be another chip's slave and helper, it's called a *support chip*.

Each chip in your computer has one of those purposes. A chip is either a processor chip or a memory chip or an interface chip or a support chip—or it's a combination chip that accomplishes *several* purposes.

CPU

WHAT DOES A CPU LOOK LIKE?

The part of the computer that thinks ("the brain") is called the *processor* (or *central processing unit* or *CPU*). It consists of one or more chips. Each chip in the processor is called a *processor chip*.

If the processor's so small that it consists of just a single chip, it's called a *microprocessor*.

If your computer's small, so is its processor. In other words, if you have a microcomputer, its brain is a microprocessor. Yes, the microcomputer's entire brain is just a single chip—a tiny square of metal!

POPULAR MICROPROCESSORS

Today, the most popular microprocessors are manufactured by four companies: MOS Technology (which is part of Commodore), Zilog (which is part of Exxon), Intel (which is partly owned by IBM), and Motorola.

You can find MOS Technology's microprocessor (which is called the 6502) inside computers sold by Commodore, Apple, and Atari. You can find Zilog's microprocessor (which is called the Z-80) inside computers sold by Radio Shack, Osborne, Kaypro, and Timex. You can find Intel's microprocessor (which is called the 8088) inside computers sold by IBM and DEC. You can find Motorola's microprocessor (which is called the 68000 and is fancier than the others!) inside the highest-priced computers sold by Radio Shack, Apple, and Fortune.

The typical microprocessor comes in a DIP (caterpillar) that has 40 pins (20 pairs of pins). Motorola's 68000 microprocessor is extra-fancy: its DIP has 64 pins instead of 40.

PRICES

How much does a microprocessor cost?

MOS Technology's 6502 microprocessor costs $5. A speeded-up version (called the 6502A) costs $7. An even faster version (called the 6502B) costs $15.

Zilog's Z-80 microprocessor costs $4. The Z-80A (which is faster) costs $5. The Z-80B (which is even faster) costs $12.

The microprocessors by Intel and Motorola cost more, because they're fancier. Intel's 8088 costs $40. Motorola's 68000 costs $60.

Those are the prices you pay at a popular electronics store (JDR Microdevices, 1224 South Bascom Ave., San Jose, CA 95128, phone 408-995-5430, out-of-state 800-538-5000). Those are the prices for "quantity one"; if you buy larger quantities, you get discounts.

Notice that a microprocessor is cheap: it costs between $4 and $60, depending on which version you get. But the computer that contains the microprocessor is much more expensive: the entire computer can cost *thousands* of dollars. That's because the microprocessor is just one tiny part of the computer. In addition to the microprocessor, you need memory chips, interface chips, and support chips; you also need PC boards to put the chips on; you also need a keyboard; you also need a tape recorder or a disk drive; you also need a printer; and you also need programs that explain to the computer what you want it to do. So even though the microprocessor can cost as little as $4, you can wind up spending $5,000 by the time you walk out of the computer store!

MEMORY CHIPS

WHY YOU NEED MEMORY

Although the CPU (the computer's brain) can think, it can't remember anything. It can't even remember what problem it was working on!

In addition to the CPU, you must buy some *memory*. The memory can remember what problem the CPU was working on. To find out what the problem was, the CPU looks at the memory frequently—about a million times every second!

You can buy two forms of memory. One form is made of chips; the other form is made of magnetized iron. Let's begin by looking at chips. . . .

PACKAGING

Like the CPU, a memory chip comes in a DIP, which is mounted on a PC board. Usually, the CPU and the memory chips both sit on the same PC board (which is called the *motherboard*). But if you buy extra memory chips (so that your computer can remember extra information), and the extra memory chips don't all fit on the motherboard, you must buy an extra PC board to mount them on. That extra PC board is called a *memory board*. If the memory board is small, it's called a *memory card*.

If your computer is ultra-modern, its memory card comes in a cute little cartridge, which is called a *memory cartridge*. You can pop the cartridge into the computer easily. Later, you can easily pull the memory cartridge out the computer and insert a different memory cartridge into the computer instead.

Warning: turn off the computer's power, before you insert a memory chip or card or cartridge. If you forget, and accidentally leave the power on while you're inserting (or removing) the memory, you might wreck your computer!

RAM CHIPS

The most popular kind of memory chip is called *random-access memory (RAM)*. The RAM chips are very flexible: you can put information into the chips, revise the information in the chips, and even erase information from the chips.

Unfortunately, RAM chips require electricity. If you turn off the computer's power, the RAM chips forget all the information you put in them.

When you buy a RAM chip, it doesn't contain any information yet: it's blank. You can put your own information into it. The RAM chip will remember your information until you erase that information or turn the power off.

BYTES IN A RAM

If you want to buy a RAM, the most important question to ask the salesman is, "How much information can I put into the RAM?"

To answer that question, the salesman will tell you how many "bytes" you can put into the RAM. A *byte* is a character; it's any symbol you can type on the keyboard. For example, it can be a letter or digit or punctuation mark or blank space. So if the salesman says the RAM can hold about 64,000 bytes, he means the RAM is large enough to hold (and remember) about 64,000 characters (symbols).

For example, the word FAT consists of 3 bytes. The name MR. POE consists of 7 bytes (M, R, the period, the space, P, O, and E). The phrase LOVE 2 KISS U consists of 13 bytes.

A *kilobyte* is about a thousand bytes. It's approximately the amount you can fit on a quarter of a typewritten page (if the page is single-spaced, with one-inch margins and elite type).

A *megabyte* is about a million bytes. It's approximately the amount you can fit in a 250-page book (if the book has single-spaced typewritten pages).

A *gigabyte* is about a billion bytes. It's approximately the amount you can fit in a library of a thousand books.

I said that a kilobyte is "about a thousand bytes", a megabyte is "about a million bytes", and a gigabyte is "about a billion bytes". Those definitions are just approximations. Actually, a kilobyte is a slightly *more* than a thousand bytes; a megabyte is slightly *more* than a million bytes; and a gigabyte is slightly *more* than a billion bytes. The precise definitions are as follows: a kilobyte is 1024 bytes; a megabyte is 1,048,576 bytes; and a gigabyte is 1,073,741,824 bytes.

That's because the computer's RAM is manufactured by a process that involves repeated doubling. If you multiply 2 times 2 times 2 times 2 times 2 times 2 times 2 times 2 times 2 times 2, you get 1024; so if you ask a company to manufacture a RAM that can hold "at least a thousand bytes", the company will manufacture a RAM that holds 1024. That's why, among manufacturers, 1024 bytes is called a kilobyte; 1024 kilobytes is called a megabyte; and 1024 megabytes is called a gigabyte.

The abbreviation for *kilobyte* is *K*. The abbreviation for *megabyte* is *meg* or *M*. The abbreviation for *gigabyte* is *G*.

In a typical microcomputer, the RAM holds "64K", which means "64 kilobytes". So it holds about 64 thousand bytes. Since a kilobyte's about a quarter of a typewritten page, the RAM holds 64 quarter-pages. So the RAM holds just 16 pages.

In a typical microcomputer, the RAM is a row of eight chips. Those eight chips, working together, hold 64K. That set of eight chips is called a *64K chip set*. Each chip in that set is called a *64K chip*; but remember that you need *eight* of those 64K chips to produce a 64K RAM! The most popular style of 64K chip is the *TI 4164*; although that style was invented by Texas Instruments, other manufacturers have copied it.

If your computer is fancy, it has *two* rows of 64K chips. Since each row is a 64K RAM, the two rows together total 128K.

If your computer is *very* fancy, it has *many* rows of 64K chips. For example, your computer might have four rows of 64K chips. Since each row is a 64K RAM, the four rows together total 256K.

On the other hand, if your computer is cheap and crummy, it contains inferior chips. For example, instead of containing a row of 64K chips, it might contain a row of 16K chips. The row of 16K chips produce a 16K RAM (which is inferior to a 64K RAM) The most popular style of 16K chip is the *TI 4116*.

If your computer is *very* cheap and crummy, it might contain a row of 4K chips instead. The row of 4K chips produces a 4K RAM.

Some computers are so cheap and crummy that they contain just 2K, or just 1K!

If your computer has very little RAM (16K or even less), you can make its RAM larger: you can buy extra RAM and attach it to your computer. To do that, you buy extra rows of RAM chips, or a RAM memory card (which contains several rows of RAM chips), or a RAM memory cartridge (which contains a RAM memory card).

You can buy RAM chips inexpensively, if you go to an electronics store. For example, you can buy a row of 64K RAM chips for $48; you can buy a row of 16K RAM chips for just $12. Those are the prices charged by JDR Microdevices (1224 South Bascom Ave., San Jose, CA 95128, phone 408-995-5430, out-of-state 800-538-5000).

RAM THAT IMITATES A DISK

RAM works faster than a disk. Suppose you buy a program that uses a disk. To make the program work faster, replace the disk by a "fake" disk that really consists of RAM.

Such a fake disk goes by several names: you can call it a *Ramdisk* (because the fake disk consists of RAM chips) or a *Semidisk* (because the RAM chips are MOS, which is metal-oxide semiconductor) or a *Solid-State Disk*. The terms *Ramdisk*, *Semidisk*, and *Solid-State Disk* are trademarked by the companies that sell them.

Although the fake disk works faster than a real disk, the fake disk has one disadvantage: since it's made of RAM, it forgets all the information when you turn the power off. (A real disk remembers information even when the power is off.)

So if you use a fake disk and want to store its information forever, you must copy the fake disk's information to a real disk before turning off the power.

ROM

Besides buying random-access memory (RAM), you can buy *read-only memory (ROM)*. The read-only memory (ROM) holds information the computer can read but can't change.

The information in the ROM remains there *permanently*. It remains in the ROM even if you turn off the power.

When you buy a ROM, it already contains information put there by the manufacturer. You can't alter that information.

When you buy a ROM, the most important question to ask the salesman is, "What information did the manufacturer put into the ROM?" For example, you can buy a ROM that contains a program that makes the computer play chess.

HOW THE COMPUTER UNDERSTANDS BASIC

If you want to write a program by using BASIC words (such as PRINT, GO, TO, INPUT, IF, and THEN), you must make sure the definitions of those words are in the memory chips.

To do that, you have two choices. The simplest choice is to buy a ROM that contains the definitions of those words. An alternative is to buy a disk that contains the definitions of those words, and then tell the computer to copy those definitions from the disk to the RAM.

If your computer lacks disk drives, the BASIC definitions are in ROM. If your computer uses CP/M (and therefore has disks), the BASIC definitions are on disk and must be copied from the disk to the RAM.

The typical microcomputer (which has disk drives but doesn't use CP/M) uses a compromise: the definitions of the most important words are in ROM, but the definitions of advanced words are on disk (and must be copied to the RAM).

TAPE TECHNOLOGY

HOW A TAPE HOLDS INFORMATION

Look at a cassette tape. You'll see that the tape is reddish brown: it's rust-colored. In fact, the tape is a strip of plastic that's covered with rust.

Rust is made of iron, which can be magnetized. To record information onto the tape, the tape recorder generates an electromagnetic field that magnetizes the rust.

On the tape, each magnetized rust particle has its own north and south pole. The patterns formed by the north and south poles are a code that stands for the information on the tape. If the tape plays ordinary music (such as Johnny Cash or Lawrence Welk), the patterns of north and south poles stand for the music. If the tape contains computer information, the patterns of north and south poles stand for the bytes.

When you play back the tape (by pressing the tape recorder's PLAY button), the recorder senses the tape's pattern of north and south poles, interprets that code, and sends music to the recorder's speaker. If the recorder is connected to a computer, the computer intercepts the music, figures out what computer information it stands for, and copies that information from the tape to the RAM.

KINDS OF TAPE

You can buy three kinds of tape systems.

The crummiest and cheapest kind of tape system is the *audio cassette*. It's an ordinary cassette, which you put into an ordinary tape recorder—the same kind of tape recorder that you listen to music on. It transmits information slowly (about a hundred bytes per second).

A slightly better (and more expensive) kind of tape system is the *digital cartridge*. It's a cartridge that you put into a special *tape drive* that's especially designed for use by a computer. It transmits information quickly (about a thousand bytes per second). The most popular digital cartridge system (because it's the cheapest) is made by Exatron, which calls it the Stringy Floppy (because it's a stringy tape that's almost as fast as a floppy disk). You can get the Stringy Floppy tape drive for just $100 from Exatron (181 Commercial St., Sunnyvale, CA 94086, phone 408-737-7111, out-of-state 800-538-8559).

The best (and most expensive) kind of tape system is the *reel-to-reel tape drive*. Instead of using a tiny cassette or cartridge, it uses a huge reel of tape: the tape is half an inch wide and half a mile long! It transmits information *very* quickly (about a hundred thousand bytes per second). That's the kind of tape system used by maxicomputers (and by large minicomputers).

CARING FOR YOUR TAPES

Do *not* put your tape near a big magnet. If you accidentally put it near a big magnet, the magnet will destroy the tape's pattern of north and south poles, and so all the information on the tape will be destroyed.

Your office contain many devices that have magnets hidden inside. If you put your tape near one of those magnetized devices, the information on the tape will be destroyed.

For example, your office probably uses paper clips. You probably store the paper clips in a little box, called a paper-clip holder. The paper-clip holder contains a magnet; it magnetizes the paper clips. If you put one of those magnetized paper clips near a computer tape, all the information on the computer tape will be destroyed!

Any large motor generates an electromagnetic field that can destroy your tape. So keep your tape away from your phone (which contains a motor); keep your tape away from your TV (whose right side contains a motor); and keep your tape away from your stereo speakers (which contain motors). Keep your tape at least four inches away from those motors.

Another way to destroy the information on a tape is to overheat it. You'll destroy the tape's information if you leave it in hot sunshine too long. So in the summer, do *not* leave your tape on the window sill! And on a hot summer day, don't leave your tape in an overheated car.

To avoid such worries, make a copy of your tape, and store the copy far away from the original tape. Then if you accidentally destroy your original tape, you can retrieve the information from the copy. The copy is called a *backup*.

DISK TECHNOLOGY

DISKS ARE LIKE TAPES

A disk is like a tape, except that a disk is round instead of long-and-thin. Like a tape, a disk is made of plastic that's covered with magnetized rust.

Like a tape, a disk should be kept at least four inches away from magnetic paper clips, telephones, the right side of televisions, and stereo speakers. Like a tape, a disk can be destroyed by an overheated window sill or an overheated car. To be safe, make a backup copy of your disk; store that copy far away from the original.

To make a backup copy easily, you should buy *two* disk drives. Put the original disk in the first drive, put a blank disk in the other drive, and tell the computer to copy all the information from the original disk to the blank disk, so that the blank disk becomes your backup disk. Because that process is much easier if you have *two* disk drives, the typical business computer system includes two drives.

To use information that's on a disk, the computer must copy the information from the disk to the RAM. The computer copies just the information it needs at the moment.

5-INCH FLOPPY DISKS

Disks come in several sizes and styles. Let's begin by analyzing the most popular kind of disk—which is the 5-inch floppy disk.

The 5-inch floppy disk was invented by Shugart Associates. The disk's diameter is approximately 5 inches. More precisely, the disk's diameter is $5\frac{1}{8}$ inches, and its square black jacket is $5\frac{1}{4}$ inches. Because the jacket's exactly $5\frac{1}{4}$ inches, some programmers call it a "$5\frac{1}{4}$-inch floppy".

To use the disk, you put it into a *disk drive*, which resembles a record player. Like a record player, the disk drive has an *arm* with a "needle" on it; the arm lowers, so that the "needle" touches the surface of the disk. The disk drive's "needle" is called the *read-write head*, since it reads the information from the disk and writes new information onto the disk.

Once the head's been lowered onto the disk, the head remains stationary, while the disk spins. As the disk spins, the head draws a circle on the disk's surface. The circle's called a *track*. To draw the circle, the head doesn't use ink; instead, it uses magnetism. Since your eye can't see magnetism, your eye can't see the circle; but it's there!

The disk spins quickly. It revolves 5 times per second (which makes 300 rpm).

When you start using a disk, the arm puts the head near the disk's outer rim, so that the head's track (circle) is almost as wide as the disk. That track is called *track 0*.

Then the arm lifts the head, moves the head slightly closer to the disk's center, and puts the head back down onto the disk again. The head draws another circular track on the disk, but this new circular track is slightly smaller than the previous one. It's called *track 1*.

Then the head draws track 2, then track 3, then track 4, and so on, until the head gets near the center of the disk, and draws the last circular track (which is smaller than the other tracks).

To organize the information on a track, the computer divides the track into *sectors*. Each "sector" is an arc of the circle.

The Apple 2e computer uses just one side of the disk; it doesn't use the flip side. It puts $\frac{1}{4}$K per sector, 16 sectors per track, and 35 tracks per disk, giving a grand total of "$\frac{1}{4}$K times 16 times 35", which is 140K. A pair of disk drives can therefore handle 280K.

The Radio Shack model 4 computer (like its predecessor, the model 3) uses just one side of the disk; it puts $\frac{1}{4}$K per sector, 18 sectors per track, and 40 tracks per disk, giving a grand total of 180K. A pair of disk drives can therefore handle 360K.

The IBM Personal Computer has changed several times during the last few months. The newest version of the IBM Personal Computer uses both sides of the disk. Yes, it has *two* read-write heads: one head uses the disk's top surface, while the other head uses the disk's bottom. The newest version (which is called "DOS version 2") puts ½K per sector, 9 sectors per track, 40 tracks per side, and 2 sides per disk, giving a grand total of 360K. A pair of disk drives can therefore handle 720K.

If your computer uses an older DOS, it puts fewer bytes on the disk. For example, although Apple's current DOS (version 3.3) puts 16 sectors per track, Apple's previous DOS (version 3.2) put only 13 sectors per track. Although Radio Shack's current DOS (used on models 3 and 4) puts 18 sectors per track and 40 tracks per disk, Radio Shack's previous DOS (which was used on model 1) put only 10 sectors per track and 35 tracks per disk. Although IBM's current DOS (version 2) puts 9 sectors per track, IBM's previous DOS (version 1) put only 8 sectors per track. Therefore, a disk for an old computer won't work on a new computer, and a disk for a new computer won't work on an old computer, unless you modify the DOS.

If a DOS puts between 75K and 150K on a side of a 5-inch disk, the DOS is called *single-density*. If it puts between 150K and 300K, it's called *double-density*. If it puts between 300K and 600K, it's called *quad-density*.

Apple's DOS (which puts 140K per side) is therefore considered single-density. Radio Shack's DOS and IBM's DOS (which put 180K per side) are considered double-density.

Since Apple's DOS uses just one side of the disk and is single-density, it's called *single-sided single-density (SSSD)*. Since Radio Shack's DOS uses just one side of the disk and is double-density, it's called *single-sided double-density (SSDD)*. Since IBM's DOS uses both sides of the disk and is double-density, it's called *double-sided double-density (DSDD)*.

If you buy a blank disk (to put into your disk drive), the disk contains no information. It contains no tracks; it contains no sectors. It's just plastic covered with rust; and the rust hasn't been magnetized yet.

After you put the blank disk into your disk drive, the next thing you must do is *format the disk*. That means you must divide the disk's surface into tracks and sectors. To do that, you type a special command, which makes the computer tell the disk drive to format the disk. After the disk has been formatted, you can start storing information on the disk.

Suppose you have an IBM Personal Computer (which uses a double-sided double-density DOS). If you walk into a computer store to buy a blank disk, the salesman will ask whether you want a "single-sided disk" or a "double-sided disk". If you request a double-sided disk, the salesman will sell you a disk that's guaranteed to work on both sides. If you request a single-sided disk instead, the salesman will sell you a cheaper disk that has been tested on only one side. The flip side of the cheap disk will *probably* work, but nobody's tested it, so the flip side isn't guaranteed.

If you buy that cheap disk, the probability is about 99% that the flip side will work. When you put it into your IBM's disk drive and try to format the disk, the computer will test the disk. If the computer discovers that the disk's flip side is grossly inferior, the computer will say so and will tell you which track on the flip side isn't working. So if the computer does *not* complain about the cheap disk, the cheap disk will probably work adequately. Moreover, if you make backup copies of all your disks, you'll never get yourself into trouble, even if you use cheap disks.

Similarly, if you request a "double-density" disk, the salesman will sell you a disk that's guaranteed to handle a double-density DOS. If instead you request a "single-density" disk, the salesman will sell you a cheaper disk that hasn't been tested for double-density but that can *probably* handle even a double-density DOS.

You can get the cheapest disks (single-sided single-density) for about $1.50 from discount mail-order companies. On the other hand, if you walk into your local computer store and request a "double-sided double-density" disk at list price, you'll pay about $8—and you're a fool!

If the disk has many tiny holes in it, the disk is called *hard-sectored*. If it has just one tiny hole in it, the disk is called *soft-sectored*. Today, most companies use soft-sectored disks.

8-INCH FLOPPY DISKS

An 8-inch floppy disk resembles a 5-inch floppy disk but is larger and therefore can hold more information.

If your DOS is old, it uses just one side of the disk. It puts an eighth of a K per sector, 26 sectors per track, and 77 tracks per disk, giving a grand total of 250¼K. A pair of such disks holds 500½K. That old format was invented by IBM.

Today, most computers use a newer format that puts more information onto the disk. For example, Radio Shack's model 12 and model 16 computers put ¼K per sector, 32 sectors per track, 77 tracks per side, and 2 sides per disk—except that track 0 of side 1 is special: that track uses the old IBM format (26 sectors, an eighth of a K per sector). Altogether, Radio Shack's disk holds 1227¼K (which is slightly more than a megabyte). A pair of disks holds 2454½K (which is slightly more than 2 megabytes).

If a DOS puts about 250K on a side of a 8-inch disk, the DOS is called *single-density*. If it puts about 500K, it's called *double-density*. IBM's old DOS is single-density; Radio Shack's DOS is double-density.

The diameter of an "8-inch" disk is actually just 7⅞. But the square jacket it's in has exactly 8 inches on each side.

An 8-inch disk rotates 6 times per second (360 rpm). By contrast, a 5-inch disk rotates only 5 times per second (300 rpm).

WRITE-PROTECT NOTCH

When you buy a 5-inch or 8-inch floppy disk, the disk comes in a square black jacket. Since the jacket's square, it has four sides; but one of the sides has a notch cut into it.

You can cover the notch, by sticking a plastic *tab* over it. The tab has a gummed back, so you can stick it on the disk easily and cover the notch. You get the tab free when you buy the disk.

Whenever you ask the computer to change the information on the disk, the computer checks whether you've covered the notch. To check, the computer makes the disk drive shine a light at the notch; a photoelectric cell determines whether the light penetrated the notch. If the light did *not* penetrate the notch, the computer knows you covered the notch with the tab.

For a 5-inch disk, the normal situation is for the notch to be uncovered. For an 8-inch disk, the normal situation is for the notch to be covered.

If the situation's normal, the computer will obey your command: it will change the information on the disk as you wish. But if the situation's *ab*normal (because the notch is covered when it should be uncovered, or is uncovered when it should be covered), the computer will *refuse* to change the disk's information.

Suppose your disk contains valuable information, and you're afraid some idiot will accidentally erase or alter that information. To prevent such an accident, make the situation abnormal (by changing whether the notch is covered), so that the computer will refuse to change the disk's information. It will refuse to erase the disk; it will refuse to add new information to the disk; it will refuse to alter the disk; it will refuse to write onto the disk. The disk is protected from being changed; it's protected from being written on. The disk is *write-protected*.

Since the tab affects whether the disk is write-protected, the tab is called a *write-protect tab*, and the notch is called a *write-protect notch*.

3-INCH FLOPPY DISKS

Recently, engineers have invented "3-inch" floppy disks.

The original "3-inch" floppy disk was invented by Hitachi. Its jacket is exactly 3 inches on each side. That size is popular in Japan, because it's small enough to fit into a Japanese envelope and be easily mailed at a Japanese post office. The 3-inch jacket is made of thick, tough plastic that cannot be bent; it protects the disk completely.

Americans prefer a jacket that's slightly larger—3½ inches on each side—because that's the width of a pocket in an American's shirt. The 3½-inch jacket was invented for Americans by Sony. Like the 3-inch jacket, it's made of tough plastic that cannot be bent. The disk uses ultra-modern technology, so that it holds as much as a double-density 8-inch disk! Yes, each side of the "3½-inch disk" holds about half a megabyte! Therefore, the entire disk holds about a megabyte; a pair of such disks holds about 2 megabytes.

HARD DISKS

Hard disks are better than floppy disks but are more expensive.

Hard disks come in three popular diameters: 5-inch, 8-inch, and 14-inch. The larger the diameter, the more bytes the disk can hold.

Maxicomputers (and the largest minicomputers) use 14-inch hard disks. Smaller computers use smaller disks.

A hard disk is either *removable* or *non-removable*. If it's *removable*, you can remove it from the disk drive. If it's *non-removable*, you must leave it in the disk drive permanently.

Although a floppy disk has a jacket, a hard disk does not. Instead of a jacket, a removable hard disk comes in a *cartridge* or *pack*.

A *cartridge* is similar to a jacket, but is round instead of square. The cartridge is made of sturdy plastic. The cartridge has a trap door that the disk drive's arm peeks through.

A *pack* is a whole stack of disks, arranged on a *spindle* that goes through their centers. The stack's covered by a clear plastic cylinder, which you can remove.

A typical pack contains 11 double-sided 14-inch disks. Each side of each disk has about 800 tracks; each track holds about 13K. The entire pack holds about 200 megabytes. It revolves 60 times per second (3600 rpm).

HEAD MOTION

A phonograph record lies horizontally on a turntable. Similarly, in most disk drives, the disk lies horizontally. But some disk drives are positioned upright, so that the disk is vertical.

Here's you get information from the top side of a typical horizontal disk. First, the disk drive's arm moves across the disk, until the read-write head (the "needle") is above the correct track of the disk.

Then, if the disk is floppy, the head descends, until it touches the disk. But if the disk is hard (instead of floppy), the head does *not* descend; instead, the head's already close enough to the disk so that it can sense the direction of the disk's magnetism.

If your hard-disk drive is ultra-modern, the head acts as as a miniature airplane: it *flies* above the disk. It flies at a very low altitude: a tiny fraction of an inch. The only thing keeping the head off the rotating disk is a tiny cushion of air—a breeze caused by the disk's motion. When you unplug the disk drive, the disk stops rotating, so the breeze stops, and the head comes to rest on a *landing strip*, which is like a miniature airport. Such a disk drive is called a *flying-head drive*.

The first flying-head drive was invented by IBM; when the drive was still top-secret, IBM's code name for it was "Winchester". Today, IBM and many other companies sell such drives, which are still called *Winchester drives*. Unfortunately, Winchester disks are non-removable.

A large Winchester drive that contains several 14-inch disks can store 300 megabytes altogether; it's used by maxicomputers. A small Winchester drive that contains 8-inch disks will store 10 megabytes; it's used by the fanciest microcomputers. A tiny Winchester drive that contains 5-inch disks will store just 5 megabytes.

To get information from a typical disk, you must wait for the arm to move its head to the right track. To eliminate such waiting, you can buy a disk drive that has *many* heads on the arm; each track has its own head. The heads and arm are stationary, and never have to move. That's called a *fixed-head disk drive*. It's super-quick, but also super-expensive, since it contains so many heads. The cheaper kind of disk drive—the kind that's *moving-head*—is more popular. All floppy-disk drives are moving-head. Only the largest, most expensive maxicomputers use fixed-head drives; such a huge computer has one or two fixed-head drives, and many, many moving-head drives. The fixed-head drive contains the information that's most important, used most often, and needed most quickly.

BUYING A PRINTER

KEYBOARD

The typical printer lacks a keyboard. (If the printer has a keyboard, it's called a *keyboard send-receive printer* or *KSR printer* or *printing terminal*.)

IMPACT PRINTERS

The typical printer works like a typewriter: a character (carved in metal or plastic) presses an inked ribbon against the paper, so that the ribbon's ink winds up on the paper. Such a printer—which presses a ribbon against the paper—is called an *impact* printer.

You can buy several kinds of impact printers.

One kind of impact printer uses a plastic *golf ball*. The character are engraved all over the golf ball (except that the very top and the very bottom of the golf ball are blank). To print a character, the golf ball rotates vertically and horizontally, until desired character is next to the ribbon; then the golf ball hits the ribbon. (That's the same method of printing used by IBM Selectric typewriters. The golf ball is also called a *Selectric typing element*. That method of printing was invented by IBM.)

A different kind of impact printer uses a *daisy wheel* instead of a golf ball. The daisy wheel looks like a daisy flower: it has petals. Each petal has a character engraved on it: for example, one petal has the letter A engraved on it; another petal has B; another has C; etc. To print a character, the daisy wheel rotates until the desired character's petal is next to the ribbon; then a hammer presses that petal against the ribbon. The daisy wheel (and its petals) can be made of either plastic or metal. Metal is more durable and produces a higher-quality impression than plastic; on the other hand, metal is more expensive, and its heaviness prevents it from moving as quickly as plastic. Most daisy-wheel printers use plastic. The first daisy-wheel printer was invented by Diablo, which is a subsidiary of Xerox; but today, daisy-wheel printers are made by dozens of other companies also.

Instead of using an ordinary daisy wheel, some printers used a "wilted" daisy, whose petals have been bent. Because the petals are bent, the "daisy" looks more like a cup. Such a device ought to be called a "wilted daisy" or a "cup"; but its inventor, who had a strange imagination, thought it looked like a gigantic thimble, and therefore called it a *thimble printer*. The inventor was a Japanese company: Nippon Electric Company. Its American subsidiary is called NEC (which is pronounced "neck"), and the printer is called the NEC Spinwriter.

A fourth kind of impact printer uses what looks like a gigantic rubber band; but the band is made of steel or plastic and has all the characters engraved on it. The band is quite large: it goes all the way from the left side of the paper to the right side. Hammers press the band against the ribbon. The printer has many hammers, which strike the band simultaneously, so that many characters are printed at once. The band moves left and right across the page, as the hammers strike. Because it has so many hammers, such a printer is very quick but also very expensive. Its high price makes it appropriate for maxicomputers or minicomputers but not for microcomputers. It's called a *band printer* or *chain printer* or *train printer*.

The most popular kind of impact printer uses needles. When the tip of a needle presses against the ribbon, a dot appears on the paper. Typically, the printer has *several* needles, so that several dots appear on the paper simultaneously. To print a character, the needles puts onto the paper a pattern of dots that looks like the character; for example, the letter L might look like this:

To produce the left edge of that L, seven needles press against the paper. The needles are all part of the *print head*, which moves from left to right across the paper. To produce the L's bottom, the print head moves to the right as the bottom needle repeatedly presses against the paper. Although seven needles are sufficient for making a capital letter (such as an L), seven needles are *not* enough for making the letters that dip below the line (g, j, p, q, and y); for those letters, you need two extra needles "below the line", giving a grand total of nine needles. The "needles" are called *wires* or *guns*. A printer that has nine wires ("needles") hitting the ribbon is called a *nine-wire impact matrix printer*. The nine-wire impact matrix printer is the most popular kind of printer. Let's look at it in more detail. . . .

The letters that dip below the line are called *descenders*. A nine-wire matrix printer can handle descenders. Before you buy a matrix printer, watch it print descenders, to see whether it can handle them well. The worst printers have fewer than nine wires, and therefore can't handle descenders.

To underline a word that contains descenders, you'd need a tenth wire; but the typical printer has only nine. See what kind of compromise *your* printer makes, if you ask it to underline a descender. If your printer is extra-fancy it has more than nine wires and can therefore handle underlined descenders.

The inventor of the impact matrix printer is unknown.

In 1979 and 1980, the most popular impact matrix printers were made in New Hampshire by Centronics. Practically every printer that Radio Shack sold was manufactured for Radio Shack by Centronics.

But Centronics' employees grew too complacent and didn't improve their printers enough to meet new competition from Japan. In 1981, Radio Shack began buying printers from several Japanese companies instead, and Centronics eventually went bankrupt.

In 1981, the most popular manufacturer of impact matrix printers was Epson, a Japanese company; and Epson has remained the most popular manufacturer ever since. For example, IBM's impact matrix printer (which IBM sells for its Personal Computer) is manufactured for IBM by Epson.

Epson became popular because it was the first company to develop a printer that had a disposable print head—so that when the print head wears out, you can throw it away and pop in a new one yourself, without needing a repairman. Also, Epson was the first company to develop a low-cost impact matrix printer whose dots look "clean and crisp" instead of looking like "fuzzy blobs".

Besides Epson, many other companies make impact matrix printers also. The companies that are *successful* are all Japanese.

So far, we've examined five kinds of printers: golf ball, daisy wheel, thimble, band, and impact matrix. Each of those printers require that something hit a ribbon. Unfortunately, "hitting" is noisy. All those printers are noisy. And eventually, the parts that hit (or get hit) will wear out.

NON-IMPACT PRINTERS

In a *non*-impact printer, nothing gets hit. So a non-impact printer is quiet and never wears out. You can buy three popular kinds of non-impact printers. . . .

An *ink-jet printer* contains a tiny hose and nozzle, which squirts ink at the page.

A *laser printer* shoots laser beams (intense light waves) at a liquid. The liquid (which is called the *developer*) contains a black powder (called the *toner*). Wherever the laser hits the liquid, the liquid deposits the black powder onto the paper. The whole process is like making a photocopy, except that it uses laser beams instead of ordinary light. The laser beams are controlled by the computer. A laser printer is incredibly fast: in less than a second, it can print several pages! The most modern kind of laser printer can print 5 pages per second.

A *thermal printer* resembles an impact matrix printer; but instead of having the needles press against a ribbon, the needles send an electrical charge to the paper. The thermal printer's paper (which is special and expensive) changes color when it receives the electrical charge.

PEN PLOTTERS

A *pen plotter* is a printer that draws by using a pen. Some pen plotters use ballpoint pens; others use felt-tip. A pen plotter might be called a "semi-impact printer", since the pen "scrapes" against the paper instead of hitting it. Whereas an impact printer is noisy (it hits and bangs), and a non-impact printer is almost silent, a pen plotter hums.

Besides drawing characters, the pen plotter can also draw curves. It can use those curves to form pretty pictures and graphs.

CHARACTER QUALITY

If your printer uses a golf ball or daisy wheel or thimble or band, the characters it produces look as nice as a typewriter's. The person who receives the document might even think it was typed by a human at a typewriter.

But if your printer uses an impact matrix or ink jet or lasers or is thermal, each character it produces is made of tiny dots. The person who receives the document will see the dots (if he stares at the document closely) and will realize it was typed by a computer, not by a human at a typewriter.

A character that's made of dots is considered to look "bad". It's considered to not be good enough for a document that tries to look "personal". A character made of dots looks "computerized" and "impersonal".

So a printer that prints dots is considered adequate for sending rough drafts and for bills, but inadequate for personal letters. Such a printer is called a *draft-quality printer* or a *business-quality printer*.

A printer that prints *fully-formed characters* (instead of dots) is called a *letter-quality printer*. Most letter-quality printers use daisy wheels, though some use golf balls or thimbles or bands or pens instead.

If a printer prints dots, but prints the dots so close together that you can hardly tell the dots apart, and so that the characters look *almost* fully-formed, and so that the document looks *almost* letter-quality, the printer is said to be *correspondence-quality*. Laser printers (which are always expensive) produce correspondence quality. Other printers that produce dots (such as impact matrix, ink-jet, and thermal) give you correspondence quality only if they're expensive.

To produce correspondence quality, the typical (expensive) nine-wire impact matrix printer begins by printing some dots; then it jerks the paper very slightly and prints more dots, so that the new dots are in-between the old dots. That act of "jerking the paper" consumes time, so that the characters unfortunately get printed slowly. When you turn on the printer, you can give it a command that tells the printer whether or not you want to it produce correspondence quality. So you choose between two modes: in *correspondence mode*, the printer produces correspondence quality by jerking the paper (slowly); in *draft mode*, the printer doesn't bother to jerk the paper, doesn't bother to print "new dots between the old dots", and produces just draft quality (quickly).

FEEDING THE PAPER

To pull the paper through the printer, the printer can use either *friction* or *pins*.

Friction is what a typewriter uses: a rubber roller grabs the paper. Unfortunately, friction is unreliable: the paper will slip slightly, especially when you get near the bottom of the sheet.

Some printers use paper that has holes in the margins. The printer has *pins* that fit in the holes. The pins move vertically, and pull the paper through the printer. The pins move the paper very accurately. The only disadvantage of pin feed is that you must buy paper having holes in the margins. The holes look ugly.

Some printers use friction feed; some use pin feed instead. If your printer's nice, it can handle friction feed and can also handle pin feed.

If your printer uses pin feed and is fancy, it has a clamp that helps the pins stay in the holes. The clamp (with its pins) is called a *tractor*. You get a tractor at each margin. A printer that has tractors is said to have *tractor feed*. If your printer is nice, you can adjust its right-hand tractor, to accommodate narrower paper (or mailing labels). If you can adjust both tractors, the printer is said to have *movable tractors*. Movable tractors are particularly helpful when you're printing mailing labels that have different widths. The typical printer has movable tractors; does *yours*?

Paper that has holes in it is called *pin-feed paper* (or *tractor-feed paper*). If that paper is fancy, its margin's perforated: after you've finished printing, you "rip off" the margin, including its ugly holes, so that what's left looks like ordinary typewriter paper.

Instead of using that trick, you might prefer the following. . . . Attach ordinary typewriter paper onto the front of pin-feed paper, by using tape or light glue. If you put that combination into your printer, the printer will print on the typewriter paper; but the typewriter paper will move accurately, because it's glued to the pin-feed paper. When you've finished printing, unglue the paper, and discard the pin-feed paper that served as the backing. What you have left is a typewritten sheet (without holes) on which the computer printed accurately (more accurately than friction feed).

Yes, you can *buy* pin-feed paper that has typewriter paper already glued to it.

WIDTH

A traditional printer makes each character a tenth of an inch wide, so that you get 10 characters per inch. That width, which is the same as an old-fashioned typewriter, is called *pica*.

Some printers make the characters narrower, so that you get 12 characters per inch. That width, which is the same as a modern office typewriter, is called *elite*.

Some printers make the characters extra-narrow, so that you get 16½ characters per inch. That width is called *condensed*.

Some printers make the characters extra-wide, so that you get just 5 characters per inch. That width is called *extended*.

Some printers make each character a different width, so that a "W" is very wide and an "i" is narrow; that's called *proportional spacing*. It look much nicer than the other kinds (pica, elite, compressed, and extended) but is more complicated to handle.

If your printer is fancy, it can handle *all* those widths: it can handle pica, elite, compressed, extended, and proportional.

Typically, the printer prints in pica (10 characters per inch), and prints 80 characters per line (which makes a total of 8 inches of typing). The paper is slightly wider than 8 inches, to allow for margins. If the printer uses friction feed, the paper is 8½ inches wide (which is the same width as ordinary typewriter paper); if the printer uses pin feed, the paper is 9½ inches wide, to permit the margins to be wide enough to include the holes.

If you tell the printer to print condensed characters (16½ per inch) instead of pica, you get 132 characters per line instead of 80.

For a few hundred dollars extra, you can buy a printer that has an extra-wide carriage, so that it can handle extra-wide paper. Whereas a traditional pin-feed printer uses paper that's 9½ inches wide,a pin-feed printer that has an extra-wide carriage can handle paper that's 14⅞ inches wide instead. By printing condensed characters on that wide paper and by making the margins as narrow as possible, some printers (such as the Epson MX-100) can print 233 characters per line.

FONTS

If your printer uses a golf ball or daisy wheel or thimble or band, you can remove the golf ball or daisy wheel or thimble or band from the printer and insert a different one instead, that has a different font (such as *italic* or *boldface*).

If your printer produces dots or uses a pen, you change the font by giving the printer a command that makes it put the dots into a different pattern.

SPEED

A printer can print characters very quickly. But when the printer reaches the end of the line, it must jerk up the paper, to get to the next line. The "jerking" occurs relatively slowly. So to find out how fast the printer can print a document, you need to know the speed at which it prints across the line, and you must also know the speed at which it jerks up the paper.

The speed at which the computer prints across the line is usually measured in *characters per second (cps)*. Usually, the printer's manufacturer will advertise the cps rating. But unfortunately, the printer's manufacturer will usually *not* advertise the speed at which the printer jerks up the paper. So the advertisement doesn't completely answer the question of how fast the document will be printed. The only way to find out how fast the document will be printed is to watch the printer yourself, while you look at a clock.

In addition to the advertised cps rating and the speed at which the printer jerks up the paper, you must also consider a variety of other subtle factors. For example, company that makes daisy-wheel printers claims a high cps rating; but the printer achieves that high cps rating only if it's printing the same character repeatedly, so that the daisy wheel didn't have to rotate!

To indent, your printer might repeatedly print spaces; but if your printer's fancier, it can "hop" quickly to the position you desire. The printer's ability to hop affects its ability to print the document quickly, but the advertised cps rating doesn't take the printer's hopping ability into account. That's another example of how the advertised cps rating doesn't truly reflect the printer's total speed.

To prove how meaningless the cps ratings are, the editors of Consumer Reports sat in front of several different printers, made all the printers print the same document, and watched the clock. The document consisted of 170 words, and had a distance of 60 characters from the left margin to the right margin. Here's how long the printers took:

Printer	Actual time	Advertised cps
Mannesman Tally MT160L in draft mode	16 seconds	160 cps
Okidata Microline 92 in draft mode	16 seconds	160 cps
Okidata Microline 82A	17 seconds	120 cps
Epson FX-80	19 seconds	160 cps (ha ha ha!)
Gemini 10	24 seconds	100 cps
Epson MX-80 F/T	31 seconds	80 cps
Radio Shack DMP-100	40 seconds	50 cps
Axiom IMP 1	44 seconds	80 cps (ha ha ha!)
Okidata Microline 92, corresp. mode	47 seconds	40 cps
Okidata Microline 80	49 seconds	80 cps (ha ha ha!)
Mannesman Tally MT160L, corr. mode	56 seconds	40 cps
Diablo 620 daisy wheel	63 seconds	20 cps
Brother HR1 daisy wheel	69 seconds	16 cps
Bytewriter daisy wheel	159 seconds	10 cps

I listed those printers from the fastest to the slowest. I put "ha ha ha!" next to the advertised cps ratings that bear no resemblance to the actual time taken.

I wish Consumer Reports had picked a different list of printers to analyze; that list of printers is *not* the list of printers that I recommend. Some of those companies (such as Radio Shack) make other printers that are much better (and cost more); and other companies make printers that cost much less yet are almost as good.

For full details about Consumer Reports' analysis, read the October 1983 issue of Consumer Reports.

PRICES

The most popular kind of printer (nine-wire impact matrix) costs about $500 and prints about 100 cps. You'll pay more if you want it to go fast or handle extra-wide paper. You'll pay less if you go to a discount dealer or if you're willing to go more slowly or if you're willing to give up correspondence mode or if you're willing to do without descenders. For example, the Gorilla Banana printer list for just $250, but it prints only 50 cps, has no correspondence mode, and has no descenders. (It prints only 50 cps because it has only one wire, which jumps up and down and try to do the work of 7 wires!)

The traditional daisy-wheel printer prints about 45 cps and costs about $2000. But since the average buyer can't afford to spend $2000, manufacturers have invented cheaper alternatives. You can buy a slowed-down version that prints about 25 cps for about $1300. You can buy an even slower version that prints only 20 cps for under $1000; in fact, Atari plans to sell a 20 cps daisy-wheel printer for just $350! Some daisy-wheel printers are even slower and cheaper.

Thimble printers print faster (55 cps) and produce better-looking characters than daisy wheels. They cost about $2000.

Golf ball printers are slow (15 cps) and need repairs frequently, but produce the most beautiful printing (when they aren't broken!).

Recently, the price of ink-jet printers has dropped to $700. People like them because they're quiet (non-impact) and easily print in many colors.

Commodore and Radio Shack both sell pen plotters for about $200. Those cheap pen plotters write with ballpoint pens on narrow paper. More expensive plotters use felt-tip pens and wider paper.

Thermal printers are cheap ($80 or even less!) but unpopular, because they require special paper.

Band printers are very fast. If each line of a document consists of 132 characters, a band printer can print anywhere from 5 to 35 lines per second; the exact speed depends on which band printer you buy. (Some models use "trains" or "chains" instead of ordinary bands.) Usually, the printer's speed is given in *lines per minute (lpm)* rather than lines per second. The speed is anywhere from 300 lpm to 2250 lpm. Band printers cost many thousands of dollars.

Laser printers are even faster. The fastest laser printers can print 20000 lpm (5 pages per second). Laser printers are very expensive. They cost between $50,000 and $400,000; the exact price depends on which model you buy and how fast it can go.

INTERFACING

To use a printer, you must run a cable from the printer to your computer. The printer's price usually does *not* include the cable, which costs about $50 extra.

In 1979 and 1980, Centronics was the most popular manufacturer of microcomputer printers. Centronics' cable contained 36 wires.

Because Centronics was so popular, many other printer companies invented printers that imitated Centronics'. Their printers used the same 36-wire cable as Centronics, and they bragged that their printers were "Centronics compatible".

Although Centronics is now a dead company, its ghost remains. Even today, printer companies brag that their printers are "Centronics compatible".

At the back of a typical microcomputer, you can find a connector, to which you can attach a 36-wire Centronics cable. If the microcomputer does indeed have that connector, its advertisements brag that it has a *Centronics printer interface*.

Today, Radio Shack's computers use a 34-pin connector, instead of a 36-pin connector. If you want to attach Radio Shack's computer to a Centronics-compatible printer, you must buy a weird cable that has 34 pins on the computer's end but has 36 pins on the computer's end.

Some companies—like Radio Shack—use slightly fewer than 36 pins. Nevertheless, they say they have a "Centronics printer interface", because they'll sell you a cable that attaches to a 36-pin printer.

The Centronics interface is an example of a *parallel* interface. Since Centronics is no longer in business, it seems archaic to say "Centronics printer interface". Instead, some companies call it the "industry-standard parallel printer interface". So if you read an ad about a computer, and the ad says that the computer has an "industry-standard parallel printer interface", it means it can handle any Centronics-compatible printer.

Unfortunately, not all printers are Centronics-compatible. Some printers use a different kind of cable instead. The most popular alternative to a Centronics-compatible interface is an *RS-232 serial interface*, which uses a cable that contains just 25 wires.

When you buy a printer and a computer, you must make sure they match. To use a Centronics-compatible printer, your computer must contain a Centronics printer interface, and you must buy a 36-wire cable. To use an RS-232 printer instead, your computer must contain an RS-232 serial interface instead, and you need a 25-wire cable.

The Centronics printer interface transmits information more quickly than the RS-232 serial interface. On the other hand, the RS-232 serial interface can transmit information over greater distances. You can even attach the RS-232 serial interface to a *modem*, which attaches to a telephone, which lets you transmit information to anywhere in the whole wide world!

MANUFACTURERS

TIMEX SINCLAIR

The cheapest complete computer you can buy is the *Timex Sinclair 1000*. It lists for just $49.95—and discount dealers charge even less! It was invented by Mr. Clive Sinclair and further improved by Timex, which is the company that manufactures it.

Clive Sinclair is British. Because Queen Elizabeth was so impressed by his computer, she made him a knight. So now his official title is *Sir* Clive Sinclair. But throughout England, his fans call him "Uncle Clive".

The computer attaches to your TV, by using an RCA cord and a switchbox. When you buy the computer, you get the RCA cord and the switchbox at no extra charge.

Unfortunately, the computer is very limited. Since it includes hardly any RAM (just 2K), your programs must be very short—unless you buy 16K of extra RAM, which unfortunately costs another $49.95. Since the computer includes hardly any ROM (just 8K), its vocabulary is very limited: for example, it doesn't understand the words DATA and READ (which are essential for doing the examples in chapter 4), and so the computer is hard to program.

Instead of having a true keyboard, the computer has a single sheet of flabby plastic on which is drawn a *picture* of a keyboard. To type a letter of the alphabet, you press the part of the plastic that shows a picture of that letter. Because the keyboard consists of just one panel of plastic, the keyboard is called a *monopanel*. It's difficult to type on.

The computer I've been describing is called the *Timex Sinclair 1000* (or *TS 1000*). Timex also manufactures a souped-up model, called the *TS 1500*, which includes a larger RAM (16K) and keyboard that has real moving keys (though the keys are made of rubber and feel awful); it costs $79.95. Unfortunately, the TS 1500's ROM still contains only 8K, so its vocabulary is still limited: it doesn't understand the words DATA and READ.

Timex also manufactures an even fancier model, called the *TS 2068*, which is excellent: it includes an even larger RAM (48K), an even better keyboard (whose keys are plastic), and a larger ROM (24K) so that its vocabulary is very large: it understands DATA and READ and advanced programming logic and even words such as CIRCLE and SOUND! Unfortunately, it's expensive: it lists for $199.95. Because of that high price, I doubt many Americans will buy it: they'll probably continue to buy from Timex's competitors instead, such as Commodore. But the British version of it will be popular in England, because of England's patriotic love of Sir Clive Sinclair, and because Commodore's computers are more expensive in England.

Here's a summary: the TS 1000 and TS 1500 are inadequate, especially because of their tiny ROM, which includes an inadequate vocabulary. The TS 2068 is fine, but its higher price makes it unattractive to Americans: if you're an American and buy it, you'll feel lonely, because most of your friends will have different computers instead.

COMMODORE

Commodore's main computers are the *VIC-20* and the *Commdore 64*. The VIC-20 costs about $85; the Commodore 64 costs about $200. To get those low prices, go to a discount department stores, such as Child World and Toys R Us and K Mart.

The Commodore 64 costs more than the VIC-20, because the Commodore 64 is better. The VIC-20 includes just 5K of RAM; the Commodore 64 includes 64K of RAM (which is why it's called the Commodore 64); the 64K of RAM lets the Commodore 64 handle longer programs than the VIC-20. The VIC-20 includes just 16K of ROM; the Commodore 64 includes 20K of ROM; the larger ROM lets the Commodore 64 handle fancier music and graphics. On your TV, the VIC-20 displays only 22 characters per line; in fact, the VIC-20 got its name because it displays "about 20" characters per line; the Commodore 64 displays 40 characters per line.

If you're buying a computer just so that you can practice writing programs, the VIC-20 is adequate. In fact, the VIC-20 is the cheapest adequate computer you can buy! (The TS 1000 and TS 1500 are cheaper but are *in*adequate.) Get the Commodore 64 only if you're rich enough to afford $200.

After buying a VIC-20 or Commodore 64, your next purchase should be either a tape recorder or disk drive. Commodore's tape recorder lists for $75, though discount department stores charge slightly less (about $65). For Commodore's disk drive, discount department stores charge about $250. Unfortunately, Commodore hasn't been able to ship enough disk drives to fill the orders, and many of the drives that Commodore *has* shipped are made crudely and soon needed realignment.

If you can afford to spend more, get Commodore's printer ($250 at discount). Although you can attach Commodore's computers to your home's color TV, you'll get a sharper picture if you use Commodore's color monitor, for which discount stores charge $250.

Commodore's computers are cheap and adequate, though they aren't quite as nice as the more expensive computers sold by Commodore's competitors.

Programming Commodore's computers is slightly awkward. (Especially awkward are the commands for handling disks, business applications, music, and graphics.) Also, Commodore's equipment is slightly unreliable and might need to be repaired soon after you buy it (especially the TV cable, power transformer, and disk drive).

Because of their awkward programming and unreliability, the VIC-20 and Commodore 64 are poor computers to run your business on. They're adequate for teaching yourself the elements of programming, though higher-priced computers are better.

I recommend Commodore's computers to schools who must run programming courses on low budgets. Also, because the VIC-20 is priced so low, you can use it as a *disposable* computer: buy the VIC-20 as a cheap experiment, to find out what a computer is; then after you've learned what the VIC-20 can and can't do, throw it away (or give it to your kid or a friend) and buy a more expensive computer that meets your needs. When you go to buy that more expensive computer (which will probably cost you several thousand dollars!), the experience you gained from playing with the $85 VIC-20 makes you a wiser shopper. Playing with the $85 VIC-20 is an excellent way to prepare, before you "taking the plunge" and spend your life savings on a larger monster.

Besides the VIC-20 and the Commodore 64, Commodore also makes higher-priced computers, which are called the *PET* and the *CBM*. But I don't recommend the PET and CBM: they aren't as good as similarly priced computers from Commodore's competitors.

TEXAS INSTRUMENTS

Texas Instruments has been in business for many years: since 1930! When it began, the company had a different name (it was called Geophysical Service Incorporated); and its first product was a device to help find underground oil. Eventually, the company changed its name to Texas Instruments (which is called "T.I.") and manufactured computer chips, pocket calculators, and minicomputers.

In 1979, Texas Instruments began selling a *micro*computer, called the *TI 99/4*, for $1150. It was terrible! Computer experts shook their heads in disgust, and wondered why anybody would spend $1150 for that piece of junk. Hardly anybody bought it, and so computer experts expected that Texas Instruments would stop trying to sell it.

But Texas Instruments didn't give up. Instead, Texas Instruments improved the computer slightly (the improved version is called the *TI 99/4A*), dropped the price dramatically (so that today the TI 99/4A costs just $99.50, instead of $1150!), and ran many TV ads, in which Bill Cosby said the TI 99/4A is terrific.

In truth, even though the TI 99/4A is better than the TI 99/4, it's still quite poor. Even though the price has dropped to $99.50, I still can't recommend it. If you're looking for a computer in that price range, I recommend instead that you buy a VIC-20 (for $85) or a Commodore 64 (for $200).

Other experts agree with me—and so do most consumers. The TI 99/4A eventually become so unpopular that TI *lost* money trying to sell it. TI stopped manufacturing it, and now gives generous discounts on the ones that are left in stock.

You should *avoid* the TI 99/4A, for three reasons. . . .

Its keyboard layout is poor. For example, the RETURN key is too small. To backspace, you must press a pair *of keys instead of hitting a single key. To type a quotation mark, you must do something weird: you must press the P key while holding down the FNCT key.*

Its version of BASIC is awkward: most of the programs in The Secret Guide to Computers will not *work on the TI 99/4A. The only way to write programs easily on the TI 99/4A is to buy TI's* Extended BASIC cartridge, *which costs $90 extra. But even if you buy that cartridge, you'll face difficulties.*

TI overcharges for the "extras" (such as disk drives, extra RAM, and programs). Even though TI has recently reduced the price of its disk drives, its price is still too high.

One argument for buying the TI 99/4A was that TI had invented so many programs for it. But lately, Atari and Commodore have been catching up.

Recently, TI has invented a fancier microcomputer, called the *TI Professional Computer*. It's wonderful! It resembles the IBM Personal Computer but is better: it displays higher-quality graphics than IBM; its wonderful keyboard is laid out much more nicely than IBM's; and in spite of those advantages it costs *less* than the IBM Personal Computer! With lots of RAM and a monitor, it costs less than $3000. Its method of programming is very similar to IBM's; but since it's slightly different from IBM's, some of the programs written for the IBM Personal Computer need to be modified in order to work on the TI Professional Computer; therefore, unfortunately, fewer programs are available for the TI Professional Computer than for the IBM Personal Computer.

ATARI

In 1972, the world's first popular video game was invented. It was called "Pong". It played ping-pong. To play it, you had to insert quarters. "Pong" was invented by a California guy named Nolan Bushnell. To market the game, he founded a company and called it "Atari Incorporated", because "Atari" is a Japanese war cry that means "beware!"

After "Pong", Atari's next successful arcade game was "Asteroids". Then came dozens of others. To play all those games, you had to insert quarters; the games were placed in arcades (and in your neighborhood bar).

In 1975, Atari invented a machine that played Pong on your home's TV.

In 1976, Warner Communications Inc. bought Atari; and so Atari became a wholly-owned subsidiary of Warner. Warner's a gigantic company: it owns Warner Brother movies and cartoons and television, Warner Cable TV, DC Comics, Superman, Mad Magazine, Lauren perfumes, Atlantic records, Elektra/Asylum Records, Knickerbocker Toy, and many other subsidiaries also!

After becoming part of Warner, in 1977 Atari invented the *Video Computer System (VCS)*, which is a machine that plays a wide variety of games on your home TV. Each game comes as a ROM cartridge. Later, companies, such as Mattel and Coleco invented machines that were similar but fancier.

In 1979, Atari began selling complete personal computers. Atari's first two computers were called the *Atari 400* and the *Atari 800*.

The Atari 400 used the same ROM and software as the Atari 800. The main difference between the 400 and the 800 was that the 400's keyboard was a monopanel (a single sheet of plastic, like the Timex Sinclair 1000), whereas the 800 had real moving keys (and was more expensive).

Atari's computers (the 400 and 800) were far ahead of their time. Of all the microcomputers being sold, Atari's had the best graphics, the best music, and the best way of editing programs. (In fact, compared to the Atari, the Apple looked pitiful!) And yet Atari's prices were *lower* than Atari's competitors (such as Apple).

But Atari made two mistakes. . . .

The first mistake was that Atari didn't hire Bill Gates to write its version of BASIC; instead, it hired the same jerk who invented Apple's DOS. Like Apple's DOS, Atari's BASIC looks simple but can't handle serious business problems.

The second mistake was that Atari believed that personal computers would be used mainly for games. Atari didn't realize that personal computers would be used mainly for business. As a result, Atari sunk lots of effort into developing spectacular games, but didn't sink enough effort into developing software and hardware for word processing, accounting, and filing.

Today, people are growing tired of playing games and want to use personal computers for tasks that are more serious. Atari hasn't prepared itself for that seriousness. Although Atari's computers could be turned into serious business machines, Atari hasn't devoted the effort necessary to make that happen. So today, consumers think of Atari as being merely a "game" computer and buy from IBM instead.

To fight back, Atari has recently been distributing a much better version of BASIC (done by Bill Gates and called *Atari Microsoft BASIC 2*) and a much better word processor (called *Atari Writer*). Also, Atari has been developing a new line of computers: they have better keyboards, larger RAMs, greater reliability, and lower prices: these new wonderful computers are called the 600 XL, the 800 XL, the 1400 XL, and the 1450 XLD.

To reduce manufacturing costs, Atari suddenly fired all the workers in its California factory and built a factory in Hong Kong to use cheaper labor. Unfortunately, Atari is having difficulty getting the Hong Kong factory to run correctly, and has temporarily stopped building the computers. When Atari finally gets the factory working, it will be "too little, too late". IBM and the Holy Trinity (Apple, Radio Shack, and Commodore) have cornered the personal-computer market, and Atari will have a hard time winning it back.

In spite of Atari's problems in Hong Kong, and in spite of the company's "game" image, Atari's new improvements deserve your careful attention.

Of all the low-cost computers on the market, Atari has the best version of BASIC. (But to get that superior version, which is called Atari Microsoft BASIC 2, you must pay $90 extra, after you buy your Atari computer.)

RADIO SHACK

Charles Tandy ran a chain of leather stores. Then he bought a chain of stores called "Radio Shack". So today, the Radio Shack chain is owned by Tandy's company and is called *Tandy's Radio Shack (TRS)*.

Under Tandy's ownership, the Radio Shack chain did well, and its profits grew 30% per year. But when America's desire for CB radios decreased, Mr. Tandy had to search for another product to maintain his 30% annual growth. He decided to try selling computers. And he hoped that by selling high-quality computers, he could change Radio Shack's shoddy image.

His dreams came true. His company's first computer (called the *TRS-80 model 1*) was low-cost but reliable. It developed a good reputation, both for itself and for the "Radio Shack" name. As a result, Radio Shack has continued to grow; today there are about 9,000 Radio Shack stores and dealers.

After the model 1 came model 2, then model 3, then model 4. Today, Radio Shack sells a wide variety of computers, which Radio Shack divides into three product "lines".

Radio Shack's cheapest line consists of the *Pocket Computers*. They look like pocket calculators: they run on batteries and have tiny LCD screens:

Model name	List price	Screen's width	RAM included
PC-4	$69.95	12 characters	½K
PC-3	$99.95	24 characters	1½K
PC-2	$199.95	26 characters	2½K

The PC-4 is manufactured for Radio Shack by Casio. The PC-3 and PC-2 are manufactured for Radio Shack by Sharp. Casio and Sharp are both Japanese.

Radio Shack's next product line consists of the *Color Computers*. They use color TV's (instead of LCD screens); they plug into the wall (instead of using batteries); they have full-size keyboards (instead of pocket-calculator keys). The Color Computers come in two versions. The cheap version, which is called the *Micro Color Computer*, has a small ROM, a 4K RAM, and costs $119.95. The expensive version, which is called the *Color Computer 2*, costs $239.95 if you get an 8K ROM and a 16K RAM; $399.95 if you get a 16K ROM and a 64K RAM. The 16K ROM contains a better version of BASIC than the 8K ROM.

Radio Shack's most expensive product line consists of the following numbered models. . . .

The model 100 includes a large LCD screen that displays eight 40-character lines. It also includes a Centronics interface, an RS-232 interface, and a 32K ROM. It costs $799 with 8K RAM, $999 with 24K RAM.

The model 4 includes a 12-inch monitor, a Centronics interface, and a 14K ROM. With 16K RAM, it costs $999; with 64K RAM and two 5-inch disk drives and an RS-232 interface, it costs $1999.

The model 4P is a portable version of the model 4. To make it portable, the monitor is 9-inch instead of 12-inch. Like the model 4, the model 4P includes a Centronics interface and a 14K ROM. It includes 64K RAM and two 5-inch disk drives and costs $1799.

The model 12 resembles the model 4, but uses 8-inch disks instead of 5-inch. The model 12 includes a 12-inch monitor, a Centronics interface, two RS-232 interfaces, and an 80K RAM. With one drive, it costs $3199; with two drives, it costs $3999.

The model 16B includes a 12-inch monitor, a Centronics interface, two RS-232 interfaces, and a 274K RAM. With one 8-inch floppy-disk drive, it costs $4999. With one 8-inch floppy-disk drive plus a 15-megabyte hard-disk drive, it costs $6999.

Those are the official list prices, but you can pay less. Radio Shack frequently has a "sale", in which Radio Shack charges less than the list price. (But when Radio Shack has a sale, don't rush: when the sale ends, the "sale" price typically becomes the new list price.) To pay even less than the "sale" price and also avoid sales tax, buy mail-order from an out-of-state Radio Shack franchised discount dealer; three of the largest and most reliable discount dealers are:

Name	City	Phone	Out-of-state WATS
Computer Plus	Littleton, Mass.	617-486-3193	800-343-8124
Micro Management	Cairo, Georgia	912-377-7120	800-841-0860
Perry Computers	Perry, Michigan	517-625-4161	800-248-3823

APPLE

Apple Computer Company was begun by two guys in California, both of whom were named "Steve".

The original Apple computer was invented by Steve Wozniak, who was an engineer at Hewlett-Packard. In 1975, he offered the plans to his boss at Hewlett-Packard, but his boss said the Apple didn't fit into Hewlett-Packard's marketing plan and suggested that Steve start his own company. So he did.

He worked with his friend, Steve Jobs. Steve Wozniak was the engineer; Steve Jobs was the businessman. Both were young: Steve Wozniak was 22; Steve Jobs was 19. Both were college drop-outs. To begin the company, Steve & Steve invested just $1300, which they got by selling a used Volkswagen Micro Bus and a used calculator. They built the first Apple computer in their garage. They sold it by word of mouth, then later by ads. The advertised price was just $666.60.

In 1977, Steve & Steve invented a better computer, called the *Apple 2*. Unlike the original Apple, the Apple 2 included a keyboard and displayed graphics in color and was more expensive ($970).

The Apple 2 might have been considered just a "toy" that displayed pretty pictures. But a surprising event in Minnesota—and another surprising event in Massachusetts—made the Apple 2 suddenly become popular and be taken seriously.

The first surprise was that the *Minnesota Educational Computer Consortium (MECC)* decided to encourage all public schools in Minnesota to buy Apples. The teachers in those schools began writing programs for the Apple. Soon, more educational programs had been written for the Apple than for any other computer.

The second surprise was that Dan Bricklin (a graduate student at Harvard) and Bob Frankston (a graduate student at MIT) wrote a program called *Visicalc*, which let the Apple easily manipulate tables of numbers for accountants. They chose the Apple because, at that time, it was the only low-cost computer that had a reliable DOS. Visicalc provided a much easier way to do financial analysis than any computer program that had ever been invented before on any computer. Financial analysts found the Apple was even more useful than the most expensive maxicomputers! The Apple was perceived as the ultimate accounting machine; accountants rushed to computer stores and bought Apples. Suddenly, Apple Computer Company became a multi-million dollar company, and the two kids who ran it (Steve & Steve) became millionaires. Several years later, Visicalc was rewritten so that it could work on other brands of computers also; but in the beginning, the terms "Visicalc" and "Apple" were almost synonymous.

Apple Computer Company tried to develop a fancier computer, called the *Apple 3*, but it was unreliable and overpriced; Apple Computer Company *lost* money by trying to sell the Apple 3. Eventually, the company improved the Apple 3's reliability and lowered the price, but the Apple 3 had already earned such a poor reputation that the company found that computer an embarrassment.

So the company went back to its Apple 2, which had been a proven success, and made some minor improvements, to create the *Apple 2-plus*. Then it made some more minor improvements, to create the *Apple 2-enhanced*, which is also called the *Apple 2e*.

Unfortunately, each time the company improved the computer it also raised the price. Today, the Apple 2e's base price is $1395; after adding a pair of disk drives and a monitor and a PC card that lets the monitor display 80 characters per line, your total cost will be over $2000.

Apple Computer Inc. also sells a more expensive computer, called the *Lisa*, which is named after Steve Jobs' illegitimate daughter. It's too expensive to be practical.

Apple's also developing a new computer called the *Macintosh*.

Compared with the newer computers invented by Apple's competitors, the Apple 2e seems primitive and overpriced. But people continue to buy the Apple 2e anyway, because of the all the educational and business programs that have been invented for it.

FRANKLIN

In Philadelphia, two guys decided to try to get rich by making a computer that imitates the Apple but costs slightly less. They named their company the *Franklin Computer Company*, because Philadelphia's patron saint (Benjamin Franklin) said that saving money is wise. To avoid offending Philadelphians, they moved the company across the Delaware River, to New Jersey.

Their first computer was called the *Franklin Ace 100*. Then came an improved version, called the *Franklin Ace 1000*. Recently, the company has developed an even further improvement, called the *Franklin Ace 1200*. It competes against the Apple 2e.

About 95% of all programs for the Apple 2e work on the Franklin Ace 1200. The Apple programs that are the *least* likely to work on the Franklin Ace 1200 are the ones that do word processing.

Apple Computer Inc. sued Franklin. Apple lost the first round of the suit. Apple won the second round, but Franklin's lawyers were tough and threatened to drag out the litigation for many years. Apple finally settled: Franklin paid Apple 2½ million dollars (for having manufactured 100,000 Aces), and agreed to change the Franklin to make it slightly less compatible with the Apple.

Some people consider Franklin to be immoral, a "rip off". On the other hand, Apple's computers are overpriced; and if competition from Franklin were eliminated, Apple's prices might be even higher.

One school system in New Jersey tried to be "moral" and buy from Apple. But the town's computer store, which sold Franklin computers, sued the Board of Education for not taking the lowest bidder. The Board of Education therefore had to buy Franklin computers instead.

COLECO

When the Connecticut Leather Company decided to start making toys instead of leather goods, the company shortened its name to "Coleco". Its first computerized toy was the *Colecovision*, which plays games on your TV.

In 1983, Coleco began selling a more ambitious product: a complete computer system, called the *Adam*. Its official list price was $695, but some discount department stores charged just $550.

The Adam includes a main keyboard that contains many keys (75 keys!), two game controllers (which include paddles and numeric keypads), an RF modulator and RCA cord (so you can attach it to your TV), a Z-80 CPU, an 80K RAM, a ROM that contains a word-processing program, a high-speed tape drive (almost as fast as a disk) that holds 500K per tape, and a daisy-wheel printer that prints nicely but slowly (just 10 characters per second). The price also includes two tapes: one contains BASIC, and the other contains a game (Buck Rogers Planet of Zoom).

Adam's version of BASIC resembles Apple's very closely. But some Apple programs won't run on the Adam, because the Adam's CPU is a Z-80A instead of a 6502, and because Adam's disk system, which costs extra, can't handle Apple's DOS. The Adam differs enough from the Apple so that Apple Computer Inc. can't sue. The Adam costs less than the Apple—and less than the Franklin.

Besides running *some* Apple software, the Adam also runs all the ROM cartridges that have been invented for the Colecovision game system.

The Adam comes with three manuals that explain how to use it. The manual that explains BASIC was a rewrite, by Coleco, of an obsolete edition of volume 1 of The Secret Guide to Computers. It included my phone number. So everybody who bought an Adam got my phone number.

I've received many calls from Adam owners. Their main complaints are that Coleco's rewrite of my manual is incomplete and incorrect, and that the tapes and tape drives are so defective that *none* of the BASIC programs work. Many owners also can't get the printer to work properly.

Informal surveys done by many people and organizations seem to indicate that about 50% of the Adams shipped are defective or stop working within 3 days; the other 50% of the Adams shipped work perfectly, and their owners are thrilled. (Coleco employees think the defect rate is just 4% instead of 50%, but lots of evidence points to the contrary.)

If you receive a defective Adam, Coleco will give you a free replacement, though many customers have received replacements that don't work.

Coleco is trying to improve the Adam. I recommend that you not buy the Adam until Coleco completes its improvements. By the time you read this, maybe the improvements will have been completed?

I have a policy of candidly answering all questions that people ask me—even if the person who's asking the question is a reporter. When reporters printed negative comments about the Adam and attributed them to me, Coleco became furious, and hired employees to rip my name, address, and phone number out of Adam's manual. Coleco and I are trying to resolve that dispute.

Unfortunately, Coleco has recently raised Adam's price by about $150. I hope the higher price will permit Coleco to improve the Adam's quality.

KAYPRO

Kaypro makes three computers.

The cheapest is the *Kaypro 2*, which costs just $1595. The computer consists of two parts. One part is the keyboard. The other part is the system unit: it includes a 9-inch green-screen monitor, two single-sided double-density 5-inch disk drives, and the computer's circuitry (including a Z-80A CPU, a 64K RAM, a Centronics interface, and an RS-232 interface). The entire computer is portable: you can attach the keyboard to the system unit, in a way so that the entire computer looks like a large attaché case. The price also includes disks that contain lots of software: the CP/M operating system, Microsoft BASIC, two word processors (Wordstar and Perfect Writer), two programs that correct your spelling (The Word Plus and Perfect Speller), two electronic spreadsheets that help you do accounting (Profit Plan and Perfect Calc), and a data-management system (Perfect Filer). To get all that for just $1595 is a tremendous bargain!

The *Kaypro 4* is more expensive ($1795). It includes better disk drives (double-sided instead of single-sided), a better electronic spreadsheet (Micro Plan, which is better than Profit Plan), and extra versions of BASIC (in addition to Microsoft BASIC, you also get CBASIC and SBASIC).

The *Kaypro 10* is the most expensive ($2795). It resembles the Kaypro 4, but replaces one of the floppy-disk drives by a 10-megabyte hard disk, and includes an extra program (Superterm, which helps your computer communicate with other computers via telephone).

Kaypro is temporarily having difficulty manufacturing the Kaypro 10, but the difficulty will probably be solved soon.

Kaypro is run by typical modern Californians. On sunny days, they go outdoors and build the computers while sitting on the sidewalk. For lunch, they run to the vegetable patch in the back yard, to gather their own salads. Their new lifestyle is food for thought—and also yields a crop of good computers.

MORROW

George Morrow was one of the first guys to design disk controllers for microcomputers. Today he sells complete microcomputer systems. His most popular systems are the *Micro Decision 3* (which costs $1599 and includes a floppy-disk drive) and the *Micro Decision 11* (which costs $2745 and includes an 11-megabyte hard disk and a 128K RAM). If you buy one of his computers, you also get lots of software at no extra charge.

His prices are almost the same as Kaypro's. His Micro Decision 3 costs $4 more than a Kaypro 2; his Micro Decision 10 costs $50 less than a Kaypro 10.

Which is better: Morrow or Kaypro? It's a toss-up. Kaypro computers have the advantage of being portable and including more software. Morrow computers have the advantage of including more hardware and higher-*quality* software.

Morrow and Kaypro are both winners. If you buy from either company, you won't go far wrong.

CROMEMCO

Cromemco is a company that resembles Morrow. Like Morrow, Cromemco makes CP/M computers and has been in the computer business for many years. "Cromemco" got its name because its founders were students at Stanford University and lived in a dormitory called "Crothers Memorial Hall".

Cromemco's cheapest computer is the *C-10 Personal Computer*, which costs $1795. It resembles Morrow's Micro Decision 3 (which costs $1599).

Cromemco also makes more expensive computers, which cost about $10,000.

EPSON

Epson is the Japanese company that makes the world's best-selling printers. It also makes two computers.

Its cheap computer is called the *HX-20 Notebook Computer*. It's portable, runs on batteries, and is about the size of a notebook. It resembles Radio Shack's model 100. It includes a built-in printer and a built-in digital tape drive, which Radio Shack lacks; on the other hand, its LCD screen is smaller than Radio Shack's and therefore displays less information.

Its fancier computer is called the *QX-10*. In addition to handling the CP/M operating system, it can also handle Epson's own operating system, which is called *Valdocs* (because it's mainly for subjecting "valuable documents" to word-processing). Valdocs is very easy to use for word processing. Although the first version of Valdocs ran too slowly to be practical, Epson has finally invented a faster version. Because the first version of Valdocs ran too slowly, computer experts looked at Valdocs cynically and didn't bother writing programs for it. Even today, the only programs that work with Valdocs are the few that Epson has written itself.

IBM

In the 1950's, IBM made maxicomputers. In the 1960's, when Digital Equipment Corporation (DEC) began selling minicomputers, IBM launched a counter-attack by selling minicomputers of its own. In the 1970's, when many companies began selling microcomputers, IBM launched a counter-attack, by selling microcomputers of its own; but IBM's first microcomputers were overpriced and unpopular.

On August 12, 1981, IBM announced a new microcomputer, called the *IBM Personal Computer*. It was an amazing bargain: its price was much lower and its quality was much higher than any of IBM's previous microcomputers. As a result, the IBM Personal Computer became an instant success. The microcomputer industry, which had been dominated by three companies (Apple, Radio Shack, and Commodore), suddenly found that IBM was a powerful fourth force that overshadowed the other three.

The IBM Personal Computer is fancier than the computers by Apple, Radio Shack, and Commodore. The IBM Personal Computer has become the new "standard" of the microcomputer industry, against which all other microcomputers are compared.

To invent the Personal Computer, IBM held a contest: three separate research teams within IBM each tried to come up with the winning design for the Personal Computer. The team that was based in Boca Raton, Florida, won the contest. (One of the losing teams came up with a lower-cost but worse computer, similar to the Xerox 820. In fact, cynics call the Xerox 820 "the rejected design for the IBM Personal Computer.")

IBM asked Bill Gates (the president of Microsoft) to invent a super-fancy version of BASIC for the IBM Personal Computer. Bill responded by invented what he called *GW BASIC*. The "GW" stands for "gee-whiz!"

IBM also asked Bill to invent a disk operating system (DOS). Bill said he wasn't in the business of inventing DOS's, and recommended that IBM contact his friend Gary Kildall instead. Gary was the head of Digital Research (the company that invented CP/M). So IBM went to Gary, to try to make him invent a version of CP/M for the IBM Personal Computer.

Gary decided to play "hard to get", in the hopes that IBM would become desperate enough to pay him lots of money. So when IBM went to visit him, he purposely was "out of the office" (even though IBM had made an appointment): instead of attending the meeting, he was flying his private plane. Unfortunately for him, his idea of playing hard-to-get backfired: IBM said "heck with you", went back to Bill, and begged Bill to invent an operating system.

Bill contacted some of his neighbors who had invented an operating system. He improved their operating system slightly and then sold the improved version to IBM. He called it *MS-DOS* (the "MS" stands for "Microsoft"); IBM renamed it *PC-DOS*.

When Gary found out, he was furious. He invented a competing operating system, called *CP/M-86*. IBM agreed to offer Gary's operating system to customers also; but IBM charged customers only $40 for PC-DOS and charged them $240 for CP/M-86, to encourage them to use PC-DOS instead. IBM's strategy worked: PC-DOS became the "standard" operating system for the IBM Personal Computer!

When IBM began selling its Personal Computer, there were hardly any programs available for it, because it was brand new. IBM turned that void into a virtue, by running an ad that told programmers to get rich by buying IBM's computer and writing programs for it.

As a result, many programmers bought the IBM Personal Computer and wrote programs for it. The best programs that had been written for the Apple and for CP/M computers were rewritten, to work on the IBM Personal Computer using PC-DOS. Now the IBM Personal Computer has a larger collection of top-quality business programs than any other computer. The programmers who wrote those programs are getting rich. And because the availability of those programs makes people want to buy an IBM Personal Computer, IBM is getting even richer!

The IBM Personal Computer costs about $3000. The exact price is anywhere from $2000 to $5000, depending on which options you get.

It's illegal to make a computer that exactly imitates the IBM Personal Computer. If a company claims to make a computer that imitates the IBM Personal Computer, the imitation is only approximate. Before you buy the imitation, ask the salesman how it differs from the IBM. If he says the imitation runs *all* IBM programs, he's lying.

On November 1, 1983, IBM announced a cheaper version of its Personal Computer. The cheaper version is nicknamed the *Peanut*; its official name is the *PC Junior*. The PC Junior has less RAM and a worse keyboard than the adult version. You should probably avoid it, and keep saving your pennies until you can afford the adult version.

DEC

Digital Equipment Corporation (DEC) was started around 1960 by an MIT graduate named Ken Olsen. The company was based on three principles:

The company would build minicomputers instead of maxicomputers—because minicomputers were quicker to build, and because the company's main stockholders were old widows who wanted a quick return on their investment before they croaked.

The company's name would contain the phrase "digital equipment" instead of the word "computer"—because in those days, most investors considered computers to be just scientific toys that weren't profitable.

The company's computers would be thin and tall, so that they'd be "taller than a man"—to make them look more impressive than humans.

DEC's first computer was the *PDP-1*. Then came the PDP-2, PDP-3, etc. Today, DEC's most advanced computer is the *VAX*, which is a souped-up minicomputer.

When Apple began to become popular, Ken's first reaction was to ignore Apple. Ken gave speeches in which he said his company would never stoop so low as to try to sell "home" computers; he said that DEC was too serious a company to sell such low-priced junk.

But finally, Ken began to realize that the country was going to be spending more money on Apples than VAX's. Ken grew worried. So he invited the heads of Apple Computer Company to visit DEC and chat about the problem. During the chat, Ken expressed an interest in buying Apple Computer Company and asked how much that would cost him. The heads of Apple said "a billion dollars". Ken said, "Get out!"

Ken decided to build a microcomputer to compete against Apple. He called it "the Apple-killer". He also nicknamed it "the Robin" (because Xerox had already invented "the Worm" to compete against Apple, and Ken wanted to compete against the Worm). The Robin was sold under the name *VT-180*. Hardly anybody bought it.

Then Ken launched a more ambitious program, to sell a series of three personal computers, called the *Rainbow*, the *Decmate 2*, and the *Professional*. The *Rainbow* resembled the IBM Personal Computer but relied CP/M-86 instead of PC-DOS. The *Decmate 2* was mainly for word processing. The *Professional* was more expensive and was mainly for communicating with DEC's larger computers (such as the VAX).

Those computers didn't sell well. The Professional was too expensive. The Decmate was too limited. The Rainbow was the only one that might have succeeded; but since its CPU was designed by Intel instead of by DEC, DEC's employees didn't take any pride in it and didn't sink enough effort into marketing it.

Instead of marketing the Rainbow aggressively, DEC just let the computer "sit and rot", and even did some things to guarantee that the computer would *not* sell well: for example, DEC charged *more* for it than IBM charged for the IBM Personal Computer; DEC's method of editing programs was worse than IBM's (because DEC didn't bother to pay Bill to invent a decent editor); DEC annoyed all its customers by forcing them to buy all their blank disks from DEC instead of from independent sources; DEC tried to discourage independent companies from selling programs for the Rainbow; and DEC's expensive television ads emphasized the computer's ability to send messages to France, even though most customers haven't the faintest interest in doing so. DEC is finally starting to "wise up", but too slowly. Ken's mind is about a year behind the rest of the microcomputer industry—and so are Ken's profits. Maybe Ken should be replaced by somebody younger who's more "with it" and doesn't take a year to react to reality. Many of Ken's vice-presidents have quit and—if Ken doesn't watch out—so will his entire company.

"Pride goeth before a fall." Ken is so proud of his magnificent VAX that he can't bear to produce a stripped-down computer that more people can afford.

CHAPTER 4

POPULAR PROGRAMMING

AIDS

DISKS

While you're working with a program, the computer keeps the program in its *random-access memory (RAM)*. In a typical microcomputer, the RAM can handle only one program at a time. If you want the computer to remember *several* programs, you can put one program in the RAM but must put the remaining programs elsewhere: put them on a disk or on a tape.

Here's how to use a disk. (I'll explain tape later.)

Before typing your program, make sure the computer and the disk have both been properly prepared. To find out how to prepare *your* computer and disk, read the appendix.

Then type your program. For example, try typing this program:

```
NEW
10 PRINT "MY DAD"
20 PRINT "IS BAD"
```

Underneath the program, try typing this:

```
SAVE "JOE"
```

That makes the computer copy the program from the RAM to the disk, and name the program JOE.

Copying from the RAM to the disk is called *saving* the program. SAVE "JOE" means: save the program and name it "JOE".

Altogether, here's what you typed:

```
NEW
10 PRINT "MY DAD"
20 PRINT "IS BAD"
SAVE "JOE"
```

(On *your* computer, the procedure might be slightly different: check the appendix.)

To make sure that the program did indeed get onto the disk, type this word:

```
FILES
```

The computer will print the names of all the programs on the disk. And one of the names it will print is JOE.

The program is now in *two* places: the original version of the program is still in the RAM, and a copy of the program is on the disk; the copy is called JOE.

If you type NEW, you'll erase the RAM; and so that program won't be in the RAM anymore. But even though the original version of the program isn't in the RAM anymore, the *copy* (which is called JOE) still remains on the disk. To prove that JOE is still on the disk, type the word FILES again; the computer will say again that JOE is one of the programs on the disk.

If you don't like the name JOE, choose a different name instead. For example you can create a program named SUE, by typing a program and then saying:

```
SAVE "SUE"
```

The name of your program can be "JOE" or "SUE" or any other brief string of letters and digits. For example, it can be "LOVER" or "POEM4U".

Pick a name that reminds you of the program's purpose. For example, if the program prints a bill to a customer, call the program "BILL"; if the program plays chess, call the program "CHESS"; if the program gives a quiz, call the program "QUIZ"; if the program tutors a human about the elements of sex, call the program "SEX"; if the program tutors a human about advanced sex, call the program "SEX2".

You've seen that the word SAVE makes the computer copy a program from the RAM to the disk. The opposite of the word SAVE is the word LOAD: it makes the computer copy a program from the *disk* to the *RAM*. For example, if you want the computer to copy JOE from the disk to the RAM, type this:

```
LOAD "JOE"
```

It tells the computer to load JOE into the RAM (from the disk). After typing that, the program is in the RAM again. To prove the program's in the RAM, type the word LIST, so that the computer will LIST the program that's in the RAM. Or, if you prefer, type the word RUN, which makes the computer RUN the program that's in the RAM.

So if JOE's on a disk, you can run it by first copying it to the RAM and then typing the word RUN, like this:

```
LOAD "JOE"
RUN
```

Instead of typing that pair of lines, you can type this single line, which is a short-cut:

```
RUN "JOE"
```

That makes the computer automatically load JOE (from the disk to the RAM) and run it.

Suppose you don't like the version of JOE that's on the disk. Here's how to revise JOE. First, copy JOE from the disk to the RAM, by saying:

```
LOAD "JOE"
```

Then examine the program, by typing:

```
LIST
```

Then revise the program, by retyping some of the program's lines, or by adding new ones. When you've finished revising the version of the program that's in the RAM, you must copy the revisions from the RAM to the disk, by saying again:

```
SAVE "JOE"
```

That command makes the computer copy the revised version of JOE onto the disk (and it also erases the previous version of JOE from the disk).

Suppose you're creating a program named TOM. Suppose TOM is so long that it takes you several hours to type. Wouldn't you be depressed if, after several hours of typing, your town suddenly had a blackout, and the computer forgot everything you'd typed?

To protect yourself against such calamities, type SAVE "TOM" every fifteen minutes. That way, if your town has a blackout, you'll lose only the few minutes of your work; the rest of your work will have already been saved on a disk.

Typing SAVE "TOM" every fifteen minutes protects you against *several* kinds of calamities. It protects you against blackouts; it protects you against "computer malfunction"; and it protects you against any careless error you might make.

Each time you say SAVE "TOM", the computer makes a copy of the program that's in the RAM, puts the copy onto a disk, and saves that copy permanently—or at least until you type SAVE "TOM" again fifteen minutes later. Even if your RAM is erased (by a blackout or computer malfunction or your carelessness), TOM will still be on the disk.

If you invent many programs and put them on the disk, the disk will someday become full. If you try to put even more programs onto the disk, and there isn't any more room on the disk for them, the computer will gripe, by printing a message such as:

```
DISK FULL
```

Someday, you'll decide that you no longer have any use for JOE, and that JOE's just wasting space on your disk. To erase JOE from the disk, type this:

```
KILL "JOE"
```

TAPES

A tape resembles a disk but is cheaper and slower.

Here's how to put a program onto a cassette tape. First, type the program. Then put a tape into the tape recorder; to do that, you might have to press the recorder's EJECT button. Rewind the tape, by pressing the recorder's REWIND button. When the tape is rewound, press the recorder's STOP button. If the recorder has a counter, press the counter's button, so the counter becomes zero.

Then press the recorder's FAST FORWARD button, until you get to a part of the tape that's blank and good. (The very beginning of the tape is bad. Wait until the counter gets to at least 10.) Then press the STOP button.

If the recorder has a counter, notice the counter's number, and write it onto the tape's label, by using a pencil.

To find out what to do next, read the appendix, which explains the peculiarities of *your* computer!

DEBUGGING

If you write your own program and run it, probably it won't work. Your first reaction will be to blame the computer; don't! The probability is 99.99% that the fault is yours. Your program contains an error. An error is called a *bug*. Your next task is to *debug* the program, which means get the bugs out.

Bugs are common; top-notch programmers make errors all the time. If you write a program that works perfectly on the first run and doesn't need debugging, it's called a *gold-star program*, and means you should have tried writing a harder one instead!

It's easy to write a program that's almost completely correct, but hard to find the little bug that's fouling it up. Most of the time you spend at the computer will be devoted to debugging.

Debugging can be fun. Hunting for the bug is like going on a treasure hunt—or solving a murder mystery. Pretend you're Sherlock Holmes. Your mission: to find the bug, and squash it!

To find the bug, use three techniques:

Inspect the program.
Trace the computer's thinking.
Shorten the program.

Here are the details. . . .

Inspect the program

Take a good, hard look at the program. If you stare hard enough, maybe you'll see the bug.

Ask the computer to help you. Make the computer print the entire program. To do that, type:

```
LIST
```

Usually, the bug will turn out to be just a typing error. Maybe you:

typed the letter O instead of a zero? or typed zero instead of the letter O?
typed I instead of 1? or typed 1 instead of I?
pressed the SHIFT key when you weren't supposed to? or forgot *to press it?*
typed an extra letter? or omitted *a letter?*
typed a line you thought you hadn't? or omitted *a line?*

Trace the computer's thinking

If you've inspected the program thoroughly, and *still* haven't found the bug, the next step is to *trace* the computer's thinking. Pretend *you* are the computer. Do what your program says. Do you find yourself printing the same wrong answers the computer printed? If so, why?

To help your analysis, make the computer print *everything* it's thinking, while it's running your program. For example, suppose your program contains lines 10, 20, 30, and 40, and uses the variables B, C, and X$. Insert these lines into your program:

```
15 PRINT "I'M AT LINE 15.  THE VALUES ARE";B;C;X$
25 PRINT "I'M AT LINE 25.  THE VALUES ARE";B;C;X$
35 PRINT "I'M AT LINE 35.  THE VALUES ARE";B;C;X$
45 PRINT "I'M AT LINE 45.  THE VALUES ARE";B;C;X$
```

Then type the word RUN. The computer will run the program again; but lines 15, 25, 35, and 45 make the computer print everything it's thinking. Check what the computer prints. If the computer prints what you expect in lines 15 and 25, but prints strange values in line 35 (or doesn't even get to line 35), you know the bug occurs before line 35 but after line 25; so the bug must be in line 30.

If your program contains hundreds of lines, you might not have the patience to insert lines 15, 25, 35, 45, 55, 65, 75, etc. Here's a short-cut. . . .

Halfway down your program, insert a line that says to print all the values. Then run your program. If the line you inserted prints the correct values, you know the bug lies underneath that line; but if the line prints *wrong* values (or if the computer never reaches that line), you know the bug lies *above* that line. In either case, you know which half of your program contains the bug. In that half of the program, insert more lines, until you finally zero in on the line that contains the bug.

Shorten the program

When all else fails, shorten the program.

Finding a bug in a program is like finding a needle in a haystack: the job is easier if the haystack is smaller. So make your program shorter: delete the last half of your program. Then run the shortened version. That way, you'll find out whether the first half of your program is working the way it's supposed to. When you've perfected the first half of your program, tack the second half back on.

Does your program contain a statement whose meaning you're not complete sure of? Check the meaning, by reading a book or asking a friend; or write a tiny experimental program that contains the statement, and see what happens when you type RUN.

Hint: before you shorten your program (or write tiny experimental ones), SAVE the original version, even though it contains a bug. After you've played with the shorter versions, retrieve the original and fix it.

The easiest way to write a long, correct program is to write a short program first, debug it, then add a few more lines, debug them, add a few more lines, debug them, etc. In other words, start with a small program, perfect it, and then gradually add perfected extras, so you *gradually* build a perfected masterpiece. If you try to compose a long program all at once—instead of building it from perfected pieces—you'll have nothing more than a master*mess*—full of bugs.

Moral: to build a large masterpiece, start will a *small* masterpiece. Putting it another way: to build a skyscraper, begin by laying a good foundation; double-check the foundation, before you start adding the walls and the roof.

EDITING

Suppose you're typing a program and make a mistake. If you notice the mistake before pressing the RETURN key, you can correct the mistake by pressing the BACKSPACE key.

But what if you notice the mistake *after* you press the RETURN key? How can you go back up to the line that contains the error, and edit it easily?

Each computer has its own way of easily editing a previous line. To find out how to edit easily on *your* computer, read the appendix.

RENUMBERING

Suppose you've written a sloppy program, like this:

```
37 PRINT "SNARL"
68 PRINT "SMIRK"
91 GO TO 37
```

If you type—

```
RENUM
LIST
```

the computer will renumber your program, so that the lines are numbered 10, 20, and 30. The computer will say:

```
10 PRINT "SNARL"
20 PRINT "SMIRK"
30 GO TO 10
```

Notice that the computer even changed the "GO TO 37", so that it became "GO TO 10"!

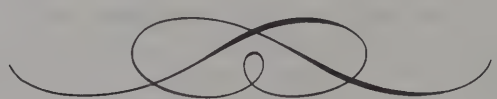
The RENUM command works on *most* computers. To find out about *your* computer, read the appendix.

PAUSING

Sometimes the computer prints information on the screen faster than you can read it. While the computer's printing, you can make the computer pause, by pressing a special pair of keys. The computer will pause, to let you read what's on the screen. After you've read what's on the screen, press another special key or pair of keys, which makes the computer continue printing, where it left off.

To find out which pair of keys makes *your* computer pause, read the appendix.

You'll want the computer to pause, if you're running a program that contains many PRINT statements, or if you're running a program that contains a PRINT statement in a loop, or if you're LISTing a long program.



APOSTROPHE

Once in a while, jot a note, to remind yourself what your program does and what the variables stand for. Slip the note into your program, by putting an apostrophe before it:

```
10 'THIS PROGRAM IS ANOTHER DUMB EXAMPLE
20 'WRITTEN BY RUSSY-POO, A STUPID JERK
30 'IT WAS WRITTEN ON HALLOWEEN, UNDER A FULL MOON, SHADED BY HIS FANGS
40 C=30' BECAUSE RUSS HAS 30 COMPUTERS
50 H=23' BECAUSE 23 OF HIS COMPUTERS ARE HAUNTED
60 PRINT C-H' THAT IS HOW MANY COMPUTERS ARE UNHAUNTED AND SAFE FOR KIDS
```

When you type RUN, the computer ignores everything that's to the right of an apostrophe. So the computer will ignore lines 10, 20, and 30; in lines 40 and 50 the computer will ignore the "because . . . "; in line 60 the computer will ignore the comments about being unhaunted. Since C is 30, and H is 23, line 60 makes the computer print:

```
 7
```

Everything that's to the right of an apostrophe is called a *comment* (or *remark*). While the computer's running the program, it ignores the comments. But whenever you say LIST, the computer will LIST the entire program, including even the comments. Although the comments appear in the LIST, they don't affect the RUN.

DATA . . . READ

LISTS OF DATA

To make the computer handle a list, begin your program with the word DATA.

For example, suppose you want the computer to print this list of food:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
```

Begin your program by saying:

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES,BUTTER,CHEESE,ONIONS,PEAS
```

You must tell the computer to READ the DATA:

```
20 READ A$
```

Line 20 makes the computer read the first datum (which is MEAT), and call it A$. So A$ is MEAT. This line makes the computer print MEAT:

```
30 PRINT A$
```

This line makes the computer handle the rest of the data:

```
40 GO TO 20
```

Line 40 makes the computer go back to line 20, which reads the next datum (POTATOES). Altogether, the program looks like this:

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES,BUTTER,CHEESE,ONIONS,PEAS
20 READ A$
30 PRINT A$
40 GO TO 20
```

Lines 20-40 form a *loop*. (Like most loops, the loop's bottom line says GO TO. Since the loop's top line says READ, the loop is called a *READ loop*.) The computer goes round and round the loop. Each time the computer comes to line 20, it reads another datum: the first time it comes to line 20, it reads MEAT; the next time, it reads POTATOES; the next time, it reads LETTUCE; the next time, it reads TOMATOES; etc. Line 30 makes the computer PRINT what it's read. Altogether, the computer will print:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
```

After the computer prints PEAS, it comes to line 40 again, which makes it go back to line 20 again, so the computer tries to read more data again; but no more data remains! So the computer says:

```
OUT OF DATA
```

Then the computer stops.

So the last three lines the computer prints are:

```
ONIONS
PEAS
OUT OF DATA
```

Instead of saying OUT OF DATA, let's make the computer say "I LIKE THOSE FOODS", so that the last three lines look like this:

```
ONIONS
PEAS
I LIKE THOSE FOODS
```

Here's how. . . . Underneath the data, say DATA END:

```
15 DATA END
```

When the computer reads the DATA END, make the computer say I LIKE THOSE FOODS and end:

```
20 READ A$: IF A$="END" THEN PRINT "I LIKE THOSE FOODS": END
```

The DATA END underneath the data is called the *end mark*, because it marks the data's end. The routine that says—

```
IF A$="END" THEN PRINT "I LIKE THOSE FOODS": END
```

is called the *end routine*, because the computer does that routine at the end.

That program prints one copy of the computer's favorite foods. If you want the computer to print *many* copies, change line 20 to this:

```
20 READ A$: IF A$="END" THEN PRINT "I LIKE THOSE FOODS": RESTORE: GO TO 20
```

The word RESTORE tells the computer to go back to the beginning of the data. The computer will print:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
I LIKE THOSE FOODS
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
I LIKE THOSE FOODS
MEAT
POTATOES
```

etc.

The computer will print copies of its list of foods again and again, forever, unless you give it an abortion.

Most practical computer programs involve a list of data. Your job, as a programmer, is to find out what the data is, and to begin your program by saying DATA. After typing the data, the next statement in your program should say READ. Farther down in your program, you should say GO TO, so that you create a READ loop. Your program consists of two parts: the DATA, and the READ loop.

You must put quotation marks around a data string, if the string contains a comma or a colon. For example, if one of the foods is "UGLY, GREEN, SLIMY, BUMPY PICKLES" (which contains commas), you must put quotation marks around it.

DEBTS

Suppose these people owe you things:

Person	What the person owes
Bob	$537.29
Mike	a dime
Frank	2 golf balls
Harry	a steak dinner at Mario's
Mommy	a kiss

Let's remind those people of their debts, by writing them letters, in this form:

```
DEAR _____
     person's name

     I JUST WANT TO REMIND YOU...

THAT YOU STILL OWE ME ____
                      debt
```

Begin with the DATA:

```
10 DATA BOB,$537.29,MIKE,A DIME,FRANK,2 GOLF BALLS
20 DATA HARRY,A STEAK DINNER AT MARIO'S,MOMMY,A KISS
```

Since the data is too long to fit on a single line, I've put part of the data in line 10, and the rest in line 20. Each line of data must begin with the word DATA. In each line, put commas between the items. Do *not* put a comma at the end of the line.

The data comes in pairs: the first pair consists of BOB and $537.29. Tell the computer to READ each pair of DATA:

```
30 READ P$,D$
```

Line 30 makes the computer read the first pair of data; so P$ is the first person (BOB), and D$ is his debt ($537.29). These lines print the letter:

```
40 PRINT "DEAR ";P$
50 PRINT "     I JUST WANT TO REMIND YOU..."
60 PRINT "THAT YOU STILL OWE ME ";D$
```

At the end of the letter, leave two blank lines, to make room for your signature:

```
70 PRINT
80 PRINT
```

Then complete the READ loop, by making the computer go back to the beginning of the loop, to read the next pair:

```
90 GO TO 30
```

The computer will print a letter to each person, like this:

```
DEAR BOB
     I JUST WANT TO REMIND YOU...
THAT YOU STILL OWE ME $537.29


DEAR MIKE
     I JUST WANT TO REMIND YOU...
THAT YOU STILL OWE ME A DIME
```

etc.

After printing all the letters, the computer will say:

```
OUT OF DATA
```

Instead of saying OUT OF DATA, let's make the computer say:

```
I'VE FINISHED WRITING THE LETTERS
```

To do that, put END at the end of the data, *twice*—

```
25 DATA END,END
```

and say what to do when the computer reaches the END:

```
30 READ P$,D$: IF P$="END" THEN PRINT "I'VE FINISHED WRITING THE LETTERS": END
```

You need *two* ENDs at the end of the data, because the READ statement says to read two strings (P$ and D$).

DIETS

Suppose you're running a diet clinic and get these results:

Person	Weight before	Weight after
Joe	273 pounds	219 pounds
Mary	412 pounds	371 pounds
Bill	241 pounds	173 pounds
Sam	309 pounds	198 pounds

Here's how to make the computer print a nice report. . . .

Begin by feeding it the DATA:

```
10 DATA JOE,273,219,MARY,412,371,BILL,241,173,SAM,309,198
```

The DATA comes in triplets: the first triplet consists of JOE, 273, and 219. Tell the computer to READ each triplet of DATA:

```
20 READ N$,B,A
```

That line makes the computer read the first triplet of data; so N$ is the first person's name (JOE), B is his weight before (273), and A is his weight after (219). These lines print the report about him:

```
30 PRINT N$;" WEIGHED";B;"POUNDS BEFORE ATTENDING THE DIET CLINIC"
40 PRINT "BUT WEIGHED ONLY";A;"POUNDS AFTERWARDS"
50 PRINT "THAT'S A LOSS OF";B-A;"POUNDS"
```

At the end of the report about him, leave a blank line:

```
60 PRINT
```

Then complete the READ loop, by making the computer go back to the loop's beginning:

```
70 GO TO 20
```

The computer will print:

```
JOE WEIGHED 273 POUNDS BEFORE ATTENDING THE DIET CLINIC
BUT WEIGHED ONLY 219 POUNDS AFTERWARDS
THAT'S A LOSS OF 54 POUNDS

MARY WEIGHED 412 POUNDS BEFORE ATTENDING THE DIET CLINIC
BUT WEIGHED ONLY 371 POUNDS AFTERWARDS
THAT'S A LOSS OF 41 POUNDS
```

etc.

At the end, the computer will say OUT OF DATA. Instead, let's make it say:

```
COME TO OUR DIET CLINIC!
```

To do that, put END and two zeros at the end of the data—

```
15 DATA END,0,0
```

and say what to do when the computer reaches the END:

```
20 READ N$,B,A: IF N$="END" THEN PRINT "COME TO OUR DIET CLINIC!": END
```

You need the two zeros after the END, because the READ statement says to read two numbers (B and A) after the string N$. If you omit the zeros, the computer will say OUT OF DATA. If you hate zeros, you can use other numbers instead; but most programmers prefer zeros.

FRENCH COLORS

Let's make the computer translate colors into French. For example, if the human says RED, we'll make the computer say the French equivalent, which is:

```
ROUGE
```

Altogether, a run will look like this:

```
RUN
WHICH COLOR INTERESTS YOU? RED
IN FRENCH, IT'S ROUGE
```

The program begins simply:

```
10 INPUT "WHICH COLOR INTERESTS YOU";C$
```

Next, we must make the computer translate the color into French. To do that, feed the computer this English-French dictionary:

English	French
white	blanc
yellow	jaune
orange	orange
red	rouge
green	vert
blue	bleu
brown	brun
black	noir

That dictionary becomes the data:

```
20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR
```

The data comes in pairs: the first pair consists of WHITE and BLANC. Tell the computer to READ each pair of DATA:

```
40 READ E$,F$
```

That line makes the computer read the first pair of data; so E$ is the first English color (WHITE), and F$ is its French equivalent (BLANC).

But that pair of data might be the *wrong pair*; for example, if the human requested RED, the human does *not* want the pair of WHITE and BLANC; instead, the human wants the pair of RED and ROUGE. So you must tell the computer: if the human's input doesn't match the English in the pair, go read another pair. Here's how to say that:

```
50 IF E$<>C$ THEN GO TO 40
```

That line says: if E$ (which is the English in the data) is not C$ (the color the human requested), go to 40 (which reads another pair).

Lines 40 and 50 form a loop; the computer goes round and round the loop, until it finds the pair of data that matches the human's request. Since the purpose of the loop is to search for data that matches, it's called a *search loop*.

After the computer has found the correct English-French pair, make the computer print the French:

```
60 PRINT "IN FRENCH, IT'S ";F$
```

Altogether, the program looks like this:

```
Ask the human→→→→→→→→→→→10 INPUT "WHICH COLOR INTERESTS YOU";C$
Use this dictionary→→→→→20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
                          30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR
Look at the dictionary→→40 READ E$,F$
If not found, try again→50 IF E$<>C$ THEN GO TO 40
Print the French→→→→→→→→60 PRINT "IN FRENCH, IT'S ";F$
```

Here's a sample run:

```
RUN
WHICH COLOR INTERESTS YOU? RED
IN FRENCH, IT'S ROUGE
```

Here's another:

```
RUN
WHICH COLOR INTERESTS YOU? BROWN
IN FRENCH, IT'S BRUN
```

Here's another:

```
RUN
WHICH COLOR INTERESTS YOU? PINK
OUT OF DATA
```

The computer says OUT OF DATA because it can't find PINK in the data. Instead of saying OUT OF DATA, it would be nicer to say I WASN'T TAUGHT THAT COLOR.

We want the computer to say I WASN'T TAUGHT THAT COLOR, when it reaches the end of the data. To do that, say END at the end of the data; and when the computer reaches the END, tell it to say I WASN'T TAUGHT THAT COLOR:

```
         10 INPUT "WHICH COLOR INTERESTS YOU";C$
         20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
         30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR
Insert→→→35 DATA END,END
Revised→→40 READ E$,F$: IF E$="END" THEN PRINT "I WASN'T TAUGHT THAT COLOR": END
         50 IF E$<>C$ THEN GO TO 40
         60 PRINT "IN FRENCH, IT'S ";F$
```

After line 60, the program just ends. Instead of letting the computer end, let's make it automatically rerun the program and translate another color. To do th , say GO TO and RESTORE:

```
  10 INPUT "WHICH COLOR INTERESTS YOU";C$
  20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
  30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR,END,END
  35 DATA END,END
→→40 READ E$,F$: IF E$="END" THEN PRINT "I WASN'T TAUGHT THAT COLOR": GO TO 70
  50 IF E$<>C$ THEN GO TO 40
  60 PRINT "IN FRENCH, IT'S ";F$
→→70 RESTORE
→→80 GO TO 10
```

FOR . . . NEXT

COUNTING TO 100

Let's make the computer print every number from 1 to 100, like this:

```
1
2
3
4
5
6
7
etc.
100
```

To do that, type this line—

```
20     PRINT X
```

and also tell the computer that you want X to be every number from 1 to 100. To tell the computer that, say "FOR X = 1 TO 100", like this:

Insert this line↔
```
10 FOR X = 1 TO 100
20     PRINT X
```

Whenever you write a program that contains the word FOR, you must say NEXT. So your program should look like this:

```
10 FOR X = 1 TO 100
20     PRINT X
```
Insert this line↔
```
30 NEXT X
```

That program works; it makes the computer print every number from 1 to 100.

Here's how it works. . . . The computer begins at line 10, which says that you want X to be every number from 1 to 100. So X starts at 1. Then the computer comes to line 20, which says to print X; so the computer prints:

```
1
```

Then the computer comes to line 30, which says to do the same thing for the next X, and for the next X, and for the next X; so the computer prints 2, and 3, and 4, and so on, all the way up to 100.

That program makes the computer print many numbers. That's because the computer does line 20 many times (once for each X).

The computer does line 20 many times, because line 20 is between the words FOR and NEXT: it's underneath FOR, and above NEXT. The computer repeats anything that's between the words FOR and NEXT.

Most programmers get in the habit of indenting everything that comes between the words FOR and NEXT; so in that program, I indented the statement that says PRINT X. To make the indentation, you can hit the space bar repeatedly.

In line 30, if you're too lazy to type the X, you can omit it and just type:

```
30 NEXT
```

PARTY

For a more advanced example, let's make the computer print these lyrics:

```
I SAW 2 MEN
MEET 2 WOMEN
TRA-LA-LA!

I SAW 3 MEN
MEET 3 WOMEN
TRA-LA-LA!

I SAW 4 MEN
MEET 4 WOMEN
TRA-LA-LA!

I SAW 5 MEN
MEET 5 WOMEN
TRA-LA-LA!

THEY ALL HAD A PARTY!
HA-HA-HA!
```

To do that, type these lines—

```
The first line of each verse→→→→→→→→→→20    PRINT "I SAW";X;"MEN"
The second line of each verse→→→→→→→→→30    PRINT "MEET";X;"WOMEN"
The third line of each verse→→→→→→→→→→40    PRINT "TRA-LA-LA!"
The blank line underneath each verse→→50    PRINT
```

and also tell the computer that you want X to be every number from 2 up to 5. Here's how:

```
Insert this line→→10 FOR X = 2 TO 5
                 20     PRINT "I SAW";X;"MEN"
                 30     PRINT "MEET";X;"WOMEN"
                 40     PRINT "TRA-LA-LA!"
                 50     PRINT
Insert this line→→60 NEXT
```

At the end of the song, print the closing couplet:

```
                 10 FOR X = 2 TO 5
                 20     PRINT "I SAW";X;"MEN"
                 30     PRINT "MEET";X;"WOMEN"
                 40     PRINT "TRA-LA-LA!"
                 50     PRINT
                 60 NEXT
Insert this line→→70 PRINT "THEY ALL HAD A PARTY!"
Insert this line→→80 PRINT "HA-HA-HA!"
```

That program works; it makes the computer print the entire song.

Here's an analysis:

```
                                                10 FOR X = 2 TO 5
                                               ⎧20     PRINT "I SAW";X;"MEN"
The computer will do these lines repeatedly,   ⎪30     PRINT "MEET";X;"WOMEN"
for X=2, X=3, X=4, and X=5.                    ⎪40     PRINT "TRA-LA-LA!"
                                               ⎩50     PRINT
                                                60 NEXT
                                               ⎧70 PRINT "THEY ALL HAD A PARTY!"
Then the computer will print this couplet once.⎩80 PRINT "HA-HA-HA!"
```

Since the computer does lines 20-50 repeatedly, those lines form a loop. Here's the general rule: the statements between FOR and NEXT form a loop. The computer goes round and round the loop, for X=2, X=3, X=4, and X=5. Altogether, it goes around the loop 4 times, which is a finite number. Therefore, the loop is *finite*.

If you don't like the letter X, choose a different letter. For example, you can choose the letter I:

```
→→10 FOR I = 2 TO 5
→→20     PRINT "I SAW";I;"MEN"
→→30     PRINT "MEET";I;"WOMEN"
  40     PRINT "TRA-LA-LA!"
  50     PRINT
  60 NEXT
  70 PRINT "THEY ALL HAD A PARTY!"
  80 PRINT "HA-HA-HA!"
```

Most programmers prefer the letter I, when using the word FOR. In other words, most programmers say "FOR I", instead of "FOR X". Saying "FOR I" is an "old tradition". I'll follow the tradition: in the rest of this book, I'll say "FOR I", except in situations where some other letter feels more natural.

SQUARES

To find the *square* of a number, multiply the number by itself. The square of 3 is "3 times 3", which is 9. The square of 4 is "4 times 4", which is 16.

Let's make the computer print the square of 3, 4, 5, etc., up to 100, like this:

```
THE SQUARE OF 3 IS 9
THE SQUARE OF 4 IS 16
THE SQUARE OF 5 IS 25
THE SQUARE OF 6 IS 36
THE SQUARE OF 7 IS 49
etc.
THE SQUARE OF 100 IS 10000
```

To do that, type this line—

```
20     PRINT "THE SQUARE OF";I;"IS";I*I
```

and also tell the computer that you want I to be every number from 3 up to 100. Here's the program:

```
10 FOR I = 3 TO 100
20     PRINT "THE SQUARE OF";I;"IS";I*I
30 NEXT
```

SECRET MEETING

This program prints 12 copies of the same message:

```
10 FOR I = 1 TO 12
20     PRINT "HUSH, HUSH!"
30     PRINT "  WE'RE HAVING A SECRET MEETING..."
40     PRINT "    IN THE COMPUTER ROOM..."
50     PRINT "      TONIGHT..."
60     PRINT "        AT 2 A.M."
70     PRINT "          WEAR A FUNNY HAT."
80     PRINT
90     PRINT
100 NEXT
```

Lines 80 and 90 leave blank lines at the end of each copy, for your signature.

MIDNIGHT

This program makes the computer count to midnight:

```
10 FOR I = 1 TO 12
20     PRINT I
30 NEXT
40 PRINT "MIDNIGHT"
```

The computer will print:

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
MIDNIGHT
```

Let's put a semicolon at the end of line 20:

```
  10 FOR I = 1 TO 12
→→20     PRINT I;
  30 NEXT
  40 PRINT "MIDNIGHT"
```

The semicolon makes the computer print each item on the same line, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12 MIDNIGHT
```

If you want the computer to press the RETURN key before MIDNIGHT, insert a PRINT line:

```
  10 FOR I = 1 TO 12
  20     PRINT I;
  30 NEXT
→→35 PRINT
  40 PRINT "MIDNIGHT"
```

Line 35 makes the computer press the RETURN key just before MIDNIGHT, so the computer will print MIDNIGHT on a separate line, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
```

In line 20, the semicolon means: do *not* press the RETURN key after I. Line 35 means: *do* press the RETURN key. So line 35 undoes line 20, and makes the computer press the RETURN key before MIDNIGHT.

Let's make the computer count to midnight 3 times, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
```

To do that, put the entire program between the words FOR and NEXT:

```
→→5 FOR A = 1 TO 3
  10     FOR I = 1 TO 12
  20         PRINT I;
  30     NEXT
  35     PRINT
  40     PRINT "MIDNIGHT"
→→50 NEXT
```

That version contains a loop inside a loop: the loop that says "FOR I" is inside the loop that says "FOR A". The A loop is called the *outer loop*; the I loop is called the *inner loop*. The inner loop's variable must differ from the outer loop's. Since we called the inner loop's variable "I", the outer loop's variable must *not* be called "I"; so I picked the letter A instead.

Often, programmers think of the outer loop as a bird's nest, and the inner loop as an egg *inside the nest*. So programmers say the inner loop is *nested in* the outer loop; the inner loop is a *nested loop*.

FAVORITE COLOR

This program plays a guessing game:

```
10 PRINT "I'LL GIVE YOU FIVE GUESSES...."
20 FOR I = 1 TO 5
30     INPUT "WHAT'S MY FAVORITE COLOR";G$
40     IF G$="PINK" THEN GO TO 100
50     PRINT "NO."
60 NEXT
70 PRINT "SORRY, YOUR FIVE GUESSES ARE UP! YOU LOSE."
80 END

100 PRINT "CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR."
110 PRINT "IT TOOK YOU";I;"GUESSES."
```

Line 10 warns the human that only five guesses are allowed. Line 20 makes the computer count from 1 to 5; to begin, I is 1. Line 30 asks the human to guess the computer's favorite color; the guess is called G$. If the guess is PINK, the computer jumps from line 40 to line 100, prints CONGRATULATIONS, and tells how many guesses the human took. But if the guess is *not* PINK, the computer proceeds from line 40 to line 50, prints NO, and goes on to the next guess. If the human guesses five times without success, the computer proceeds from line 60 to line 70 and prints SORRY...YOU LOSE.

For example, if the human's third guess is PINK, the computer prints:

```
CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR.
IT TOOK YOU 3 GUESSES.
```

If the human's very first guess is PINK, the computer prints—

```
CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR.
IT TOOK YOU 1 GUESSES.
```

—which is ungrammatical but understandable.

Lines 20-60 form a loop. Line 20 says the loop will normally be done five times. The line after the loop, line 70, is the loop's *normal exit*. But if the human happens to input PINK, the computer jumps out of the loop early, to line 100, which is the loop's *abnormal exit*.

STEP

The FOR statement can be varied:

Statement	Meaning
`FOR I = 5 TO 17 STEP .1`	I will be every number from 5 to 17, counting by tenths. So I will be 5, then 5.1, then 5.2, etc., up to 17.
`FOR I = 5 TO 17 STEP 3`	I will be every third number from 5 to 17. So I will be 5, then 8, then 11, then 14, then 17.
`FOR I = 17 TO 5 STEP -3`	I will be every third number from 17 down to 5. So I will be 17, then 14, then 11, then 8, then 5.

To count down, you must use the word STEP. To count from 17 down to 5, give this instruction:

```
FOR I = 17 TO 5 STEP -1
```

This program prints a rocket countdown:

```
10 FOR I = 10 TO 1 STEP -1
20     PRINT I
30 NEXT
40 PRINT "BLAST OFF!"
```

The computer will print:

```
 10
 9
 8
 7
 6
 5
 4
 3
 2
 1
BLAST OFF!
```

This statement is tricky:

```
FOR I = 5 TO 16 STEP 3
```

It says to start I at 5, and keep adding 3 until I gets past 16. So I will be 5, then 8, then 11, then 14. I won't be 17, since 17 is past 16. The first value of I is 5; the last value is 14.

In the statement FOR I = 5 TO 16 STEP 3, the *first value* or *initial value* of I is 5, the *limit value* is 16, and the *step size* or *increment* is 3. The I is called the *counter* or *index* or *loop-control variable*. Although the limit value is 16, the *last value* or *terminal value* is 14.

Programmers usually say "FOR I", instead of "FOR X", because the letter I reminds them of the word *index*.

FANCY PRINTING

ZONES

If your computer's fancy, the leftmost part of the screen is called the first *zone*; to the right of that zone lies the second zone; to the right of the second zone lies the third zone; then comes the fourth zone; and the rightmost part of the screen is called the fifth zone. So altogether, there are 5 zones. Each zone is 16 characters wide; the entire screen is 80 characters wide.

(If your computer's *not* so fancy, it might have fewer zones, or the zones might be narrower; and the screen might be less than 80 characters wide. To find out about *your* computer's peculiarities, read the appendix.)

A comma makes the computer jump to a new zone; here's an example:

```
PRINT "SIN","KING"
```

The computer will print SIN and KING on the same line; but because of the comma before KING, the computer will print KING in the second zone, like this:

```
SIN             KING
```

first zone *second zone* *third zone* *fourth zone* *fifth zone*

Of course, you can turn that example into a program, by putting a number in front of the line, like this:

```
10 PRINT "SIN","KING"
```

When you type RUN, the computer will print:

```
SIN             KING
```

This program does the same thing:

```
10 PRINT "SIN",
20 PRINT "KING"
```

Line 10 makes the computer print SIN and then jump to the next zone. Line 20 makes the computer print KING. The computer will print:

```
SIN             KING
```

This example's silly:

```
PRINT "LOVE","CRIES","OUT"
```

The computer will print LOVE in the first zone, CRIES in the second zone, and OUT in the third zone, like this:

```
LOVE            CRIES           OUT
```

This example's even sillier:

```
PRINT "LOVE","CRIES","OUT","TO","ME","AT","NIGHT"
```

The computer will print LOVE in the first zone, CRIES in the second, OUT in the third, TO in the fourth, ME in the fifth, and the remaining words below, like this:

```
LOVE            CRIES           OUT             TO              ME
AT              NIGHT
```

This program makes the computer greet you:

```
10 PRINT "HI",
20 GO TO 10
```

The computer will print HI many times; each time will be in a new zone, like this:

```
HI              HI              HI              HI              HI
HI              HI              HI              HI              HI
HI              HI              HI              HI              HI
etc.
```

This program prints a list of words and their opposites:

```
10 PRINT "GOOD","BAD"
20 PRINT "BLACK","WHITE"
30 PRINT "GRANDPARENT","GRANDCHILD"
40 PRINT "HE","SHE"
```

Line 10 makes the computer print GOOD, then jump to the next zone, then print BAD. Altogether, the computer will print:

```
GOOD            BAD
BLACK           WHITE
GRANDPARENT     GRANDCHILD
HE              SHE
```

The first zone contains a column of words; the second zone contains the opposites. Altogether, the computer's printing looks like a table. So whenever you want to make a table, use zones, by putting commas in your program.

Let's make the computer print this table:

```
NUMBER          SQUARE
 3               9
 4               16
 5               25
 6               36
 7               49
 8               64
 9               81
 10              100
```

Here's the program:

```
10 PRINT "NUMBER","SQUARE"
20 FOR I = 3 TO 10
30     PRINT I,I*I
40 NEXT
```

Line 10 prints the word NUMBER at the top of the first column, and the word SQUARE at the top of the second. Line 20 says I goes from 3 to 10; to begin, I is 3. Line 30 makes the computer print:

```
 3               9
```

Line 40 makes the computer do the same thing for the next I, and for the next I, and for the next; so the computer prints the whole table.

This example tells a bad joke:

```
PRINT "I THINK YOU ARE UGLY","I'M JOKING"
```

The computer will print I THINK YOU ARE UGLY, then jump to a new zone, then print I'M JOKING, like this:

You can make the computer skip over a zone:

```
PRINT "JOE"," ","LOVES SUE"
```

The computer will print JOE in the first zone, a blank space in the second zone, and LOVES SUE in the third zone, like this:

You can type that example even more briefly, like this:

```
PRINT "JOE",,"LOVES SUE"
```

When you combine commas with semicolons, you can get weird results:

```
PRINT "EAT","ME";"AT";"BALLS","NO";"W"
```

That line contains commas and semicolons. A comma makes the computer jump to a new zone, but a semicolon does *not* make the computer jump. The computer will print EAT, then jump to a new zone, then print ME and AT and BALLS, then jump to a new zone, then print NO and W. Altogether, the computer will print:

```
EAT             MEATBALLS       NOW
```

TAB

When the computer puts a line of information on your screen, the leftmost character in the line is said to be at *position 1*; the second character in the line is said to be at *position 2*. This command makes the computer skip to position 6 and then print "HOT":

```
PRINT TAB(6)"HOT"
```

The computer will print:

```
     HOT
12345678
```

Here's a fancier example:

```
PRINT TAB(6)"HOT";TAB(13)"BUNS"
```

The computer will skip to the 6th position, then print "HOT", then skip to the 13th position, then print "BUNS":

```
     HOT    BUNS
123456789101112131415 16
```

This program prints a diagonal line:

```
10 FOR I = 1 TO 12
20     PRINT TAB(I)"*"
30 NEXT
```

Line 10 says to do the loop twelve times; so the computer does line 20 repeatedly. The first time the computer does line 20, the I is 1, so the computer prints an asterisk at position 1:

```
*
```

The next time, the I is 2, so the computer skips to position 2 and prints an asterisk:

```
 *
```

The next time, the I is 3, so the computer skips to position 3 and prints an asterisk:

```
  *
```

Altogether, the program makes the computer print this picture:

```
*
 *
  *
   *
    *
     *
      *
       *
        *
         *
          *
           *
            *
```

CALENDAR

Let's make the computer print a calendar for the whole year. To begin, we must tell the computer how many days are in each month:

Month's name	Month's length
January	31
February	28 or 29
March	31
April	30
May	31
June	30
July	31
August	31
September	30
October	31
November	30
December	31

That list becomes our data:

```
10 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30
20 DATA MAY,31,JUNE,30,JULY,31,AUGUST,31
30 DATA SEPTEMBER,30,OCTOBER,31,NOVEMBER,30,DECEMBER,31
```

(For a leap year, change line 10.)

To make the computer look at that data, tell the computer to READ:

```
40 READ N$,L
```

That makes the computer read a month's Name and Length.

To read *all* the data, the computer should do line 40 twelve times. So put line 40 inside a loop, like this:

```
                10 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30
Here are the months
and their lengths.
                20 DATA MAY,31,JUNE,30,JULY,31,AUGUST,31
                30 DATA SEPTEMBER,30,OCTOBER,31,NOVEMBER,30,DECEMBER,31

For each month,
                39 FOR M = 1 TO 12

read its Name & Length,
                40     READ N$,L

print its Name,
                50     PRINT N$

print all its days,
                60     FOR D = 1 TO L
                70         PRINT D;
                80     NEXT

and press the RETURN key
twice at end of month.
                90     PRINT
                91     PRINT
               100 NEXT
```

The computer will print a calendar beginning like this:

```
JANUARY
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
 21  22  23  24  25  26  27  28  29  30  31

FEBRUARY
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
 21  22  23  24  25  26  27  28

MARCH
 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20
 21  22  23  24  25  26  27  28  29  30  31
```

Although the computer's printed the right numbers for each month, the numbers are in the wrong places. Let's make the computer print at most 7 numbers in each row, so each is a week.

To print the numbers in the right places, use TAB:

```
70          PRINT TAB(T)D;
```

Line 70 will make the computer print each day in the right position . . . if we define T correctly. But how should we define T?

For Sunday, let's make T be 2 (so that Sunday begins at position 2); for Monday, let's make T be 6 (so that Monday begins at position 6); for Tuesday, let's make T be 10; for Wednesday, 14; Thursday, 18; Friday, 22; and Saturday, 26. So whenever a day's been printed, T should normally increase by 4, for the next day:

```
71          T=T+4
```

Saturday's the last day of the week; after Saturday, we must begin a new week. So if T has passed Saturday (which is 26), we want T to become 2 (for Sunday); and if there are more days left in the month, we want the computer to press the RETURN key (to start a new week):

```
72          IF T>26 THEN T=2: IF D<L THEN PRINT
```

Which year would you like a calendar for: 1984? 1985? 1986? This program makes a pretty calendar for 1985:

```
→→1 PRINT "CALENDAR FOR 1985"
→→2 PRINT
→→3 T=10
  10 DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30
  20 DATA MAY,31,JUNE,30,JULY,31,AUGUST,31
  30 DATA SEPTEMBER,30,OCTOBER,31,NOVEMBER,30,DECEMBER,31
  39 FOR M = 1 TO 12
  40     READ N$,L
  50     PRINT N$
→→51     PRINT "  SUN MON TUE WED THU FRI SAT"
  60     FOR D = 1 TO L
  70         PRINT TAB(T)D;
  71         T=T+4
  72         IF T>26 THEN T=2: IF D<L THEN PRINT
  80     NEXT
  90     PRINT
  91     PRINT
 100 NEXT
```

Line 1 prints the heading. Line 2 puts a blank line underneath the heading. Since 1985 begins on a Tuesday, line 3 tells the computer to start T at 10 (which is the position for Tuesday). Line 51 prints the heading for each month; when you type that line, put 2 blank spaces before SUN. The computer will print a calendar beginning like this:

```
CALENDAR FOR 1985

JANUARY
  SUN MON TUE WED THU FRI SAT
            1   2   3   4   5
    6   7   8   9  10  11  12
   13  14  15  16  17  18  19
   20  21  22  23  24  25  26
   27  28  29  30  31

FEBRUARY
  SUN MON TUE WED THU FRI SAT
                        1   2
    3   4   5   6   7   8   9
   10  11  12  13  14  15  16
   17  18  19  20  21  22  23
   24  25  26  27  28

MARCH
  SUN MON TUE WED THU FRI SAT
                        1   2
    3   4   5   6   7   8   9
   10  11  12  13  14  15  16
   17  18  19  20  21  22  23
   24  25  26  27  28  29  30
   31
```

If you want a different year, change lines 1 and 3; and if you want a leap year, change line 10. If you want the calendar to be taller, insert extra blank lines; to do that, replace "PRINT" by "PRINT: PRINT: PRINT" in lines 2, 72, 90, and 91:

```
2 PRINT: PRINT: PRINT
72         IF T>26 THEN T=2: IF D<L THEN PRINT: PRINT: PRINT
90     PRINT: PRINT: PRINT
91     PRINT: PRINT: PRINT
```

If you want the calendar to look wider (or narrower), change the positions for Sunday, Monday, Tuesday, etc., by changing the T numbers (in lines 3, 51, 71, and 72).

LOCATE

The computer makes the screen show several lines of information. On the screen, the top line is called *line 1*; underneath it is *line 2*; then comes *line 3*; etc.

Each line consists of many characters. The leftmost character is at *position 1*; the next character is at *position 2*; etc.

On the screen, the computer will print wherever you wish. For example, to make the computer print the word DROWN, so that the word DROWN begins at line 3's 7th position, type this:

```
LOCATE 3,7: PRINT "DROWN"
```

The computer will print the word's first letter (D) at line 3's 7th position. The computer will print the rest of the word afterwards.

You'll see the first letter (D) at line 3's 7th position, the next letter (R) at the next position (line 3's 8th position), the next letter (O) at the next position (line 3's 9th position), etc.

Most computers understand the word LOCATE, but *your* computer might be different. To find out about *your* computer, check the appendix.

ELEMENTS

GRAMMAR

A *numeric constant* is a simple number, such as:

```
0     1     2     8     43.7     -524.6     .003     1.3E5
                                                      ↑
```

which means, "take 1.3, and move the decimal point 5 places to the right"

A numeric constant does not contain any arithmetic. For example, 7+1 contains arithmetic (+), so it is *not* a numeric constant. 8 is a numeric constant, even though 7+1 is not.

A *string constant* is a simple string, in quotation marks:

```
"I LOVE YOU"     "76 TROMBONES"     "GO AWAY!!!"     "XYPW EXR///746"
```

A *constant* is a numeric constant or a string constant:

```
0     8     -524.6     1.3E5     "I LOVE YOU"     "XYPW EXR///746"
```

A *string variable* is a variable that stands for a string:

```
A$     B$     Y$     Z$
```

A *numeric variable* is a variable that stands for a number:

```
A     B     Y     Z
```

A *variable* is a string variable or a numeric variable:

```
A$     Z$     A     Z
```

A *numeric expression* is a numeric constant (such as 8) or a numeric variable (such as A) or a combination of them, such as 8+A, or 8*A, or Z*A, or 8*2, or 7+1, or even Z*A-(7+Z)/8+1.3E5*(-524.6+B). A *string expression* is a string constant (such as "I LOVE YOU") or a string variable (such as A$) or a combination. An *expression* is a numeric expression or a string expression.

In a DATA statement, you must have constants.

```
Right:  10 DATA 8, 1.3E5        (8 and 1.3E5 are constants.)
Wrong:  10 DATA 7+1, 1.3E5      (7+1 is not a constant.)
```

In the DATA statement, if the constant is a string, you can omit the quotation marks (unless the string contains a comma or a colon).

```
Right:       10 DATA "JOE","MARY"
Also right:  10 DATA JOE,MARY
```

The INPUT statement's prompt must be a string constant.

```
Right:  10 INPUT "WHAT IS YOUR NAME";N$    ("WHAT IS YOUR NAME" is a string constant.)
Wrong:  10 INPUT Q$;N$                     (Q$ is not a string constant.)
```

In a GO TO statement, the line number must be a numeric constant.

```
Right:  50 GO TO 100     (100 is a numeric constant.)
Wrong:  50 GO TO N       (N is not a numeric constant.)
```

Here are the forms of the most popular BASIC statements:

General form	Example
PRINT list of expressions and commas and semicolons and tabs	PRINT TAB(6)"HOT",2+2;"U"
INPUT string constant ; variable	INPUT "WHAT IS YOUR NAME";N$
variable = expression	X = 47+2
GO TO numeric constant	GO TO 10
STOP	STOP
IF condition THEN list of statements	IF A>=18 THEN PRINT "YOU": PRINT "CAN VOTE"
DATA list of constants	DATA JOE,273,219,MARY,412,371
READ list of variables	READ N$,B,A
RESTORE	RESTORE
FOR numeric variable = numeric expression TO numeric expression STEP numeric expression	FOR I = 59+1 TO 100+N STEP 2+3
NEXT numeric variable	NEXT I

A VARIABLE IS A BOX

The computer's random-access memory (RAM) consists of electronic boxes. This program puts a number into a box:

```
10 X=47
20 PRINT X+2
```

Line 10 puts 47 into box X, like this:

Box X [47]

Line 20 says to print what's in box X, plus 2. So the computer will print 49.

You can change what's in a box:

```
10 F=4
20 F=9
30 PRINT F*2
```

Line 10 puts 4 into box F:

Box F [4]

Line 20 puts 9 into box F; the 9 replaces the 4:

Box F [9]

Line 30 prints 18.

Saying "A=B" has a different effect than "B=A". Compare these programs:

```
10 A=1          10 A=1
20 B=7          20 B=7
30 A=B          30 B=A
40 PRINT A+B    40 PRINT A+B
```

In both programs, lines 10 and 20 do this:

Box A [1]
Box B [7]

In the left program, line 30 makes the number in box A become 7, so both boxes contain 7; line 40 prints 14. In the right program, line 30 makes the number in box B become 1, so both boxes contain 1; line 40 prints 2.

Try this one-line program:

```
10 PRINT R
```

That line says to print a *number*— the number in box R. Since box R is empty, the computer prints this number:

```
 0
```

ARTS

RANDOM NUMBERS

Usually, the computer is predictable: it does exactly what you say. But sometimes, you want the computer to be *un*predictable. For example, if you want to play a game of cards with the computer, and you want the computer to deal, you want the cards dealt to be unpredictable. If the cards were predictable— if you could figure out exactly which cards you'd be dealt, and which cards the computer would be dealt— the game would be boring.

In many other kinds of games too, you want the computer to be unpredictable, to "surprise" you. Without an element of surprise, the game would be boring.

And if you want the computer to act artistic, and create a new *original* masterpiece that's a "work of art", you need a way to make the computer get a "flash of inspiration". And flashes of inspiration are not predictable; they are surprises.

This program makes the computer print an unpredictable number from 1 to 5:

```
10 RANDOMIZE
20 PRINT RND(5)
```

Unpredictable numbers are called *random* numbers. Line 10 tells the computer that you want the numbers to be *completely* unpredictable, *completely* random. Line 20 makes the computer print a random number from 1 to 5. So the computer will print 1 or 2 or 3 or 4 or 5; you can't predict *which* of those numbers the computer will print; the computer's choice will be a surprise.

For example, when you type RUN, the computer might print 3. If you run the program a second time, the computer might print a different number (1 or 2 or 4 or 5), or it might print the same number (3). You can't predict which number the computer will print. The only thing you can be sure of is: the number will be from 1 to 5.

(For *your* computer, you might have to write the program slightly differently. To find out your computer's peculiarities, check the appendix.)

To make the computer print *many* such random numbers, say GO TO:

```
        10 RANDOMIZE
        20 PRINT RND(5)
New line↔30 GO TO 20
```

The computer will print many numbers, like this:

```
3
2
4
4
1
3
5
2
2
5
etc.
```

Each number will be 1 or 2 or 3 or 4 or 5. The order in which the computer prints the numbers is unpredictable. The program is an infinite loop; to stop it, you must give an abortion. If you run the program again, the pattern will be different; for example, it might be:

```
1
4
3
3
2
5
1
1
2
etc.
```

That program consumes a lot of paper (or a lot of your screen). To use less paper (or less of your screen), make the computer print the numbers *across*, instead of down; make the computer print like this:

```
3  2  4  4  1  3  5  2  2  5  etc.
```

To do that, put a semicolon in the PRINT statement:

```
        10 RANDOMIZE
New line↔20 PRINT RND(5);
        30 GO TO 20
```

That program prints random numbers up to 5. To see random numbers up to 1000, say RND(1000):

```
        10 RANDOMIZE
New line↔20 PRINT RND(1000);
        30 GO TO 20
```

The computer will print something like this:

```
485  729  8  537  1000  13  1  842  842  156  1000  972  etc.
```

This program plays a guessing game:

```
10 RANDOMIZE
20 PRINT "I'M THINKING OF A NUMBER FROM 1 TO 100."
30 C=RND(100)
40 INPUT "WHAT DO YOU THINK MY NUMBER IS";G
50 IF G<C THEN PRINT "YOUR GUESS IS TOO LOW.": GO TO 40
60 IF G>C THEN PRINT "YOUR GUESS IS TOO HIGH.": GO TO 40
70 PRINT "CONGRATULATIONS!  YOU FOUND MY NUMBER!"
```

Line 20 makes the computer say:

```
I'M THINKING OF A NUMBER FROM 1 TO 100.
```

Line 30 makes the computer think of a random number from 1 to 100; the computer's number is called "C". Line 40 asks the human to guess the number; the guess is called "G". If the guess is less than the computer's number, line 50 makes the computer say "YOUR GUESS IS TOO LOW" and then GO TO 40, which lets the human guess again. If the guess is *greater* than the computer's number, line 60 makes the computer say "YOUR GUESS IS TOO HIGH" and then GO TO 40. When the human guesses correctly, the computer arrives at line 70, which prints:

```
CONGRATULATIONS!  YOU FOUND MY NUMBER!
```

Here's a sample run:

```
RUN
I'M THINKING OF A NUMBER FROM 1 TO 100.
WHAT DO YOU THINK MY NUMBER IS? 54
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 73
YOUR GUESS IS TOO HIGH.
WHAT DO YOU THINK MY NUMBER IS? 62
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 68
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 70
YOUR GUESS IS TOO HIGH.
WHAT DO YOU THINK MY NUMBER IS? 69
CONGRATULATIONS!  YOU FOUND MY NUMBER!
```

This program makes the computer roll a pair of dice:

```
10 RANDOMIZE
20 PRINT "I'M ROLLING A PAIR OF DICE"
30 A=RND(6)
40 PRINT "ONE OF THE DICE SAYS:";A
50 B=RND(6)
60 PRINT "THE OTHER SAYS:";B
70 PRINT "THE TOTAL IS";A+B
```

Line 20 makes the computer say:

```
I'M ROLLING A PAIR OF DICE
```

Each of the dice has 6 sides. Lines 30 and 40 roll one of the dice, by picking a number from 1 to 6. Lines 50 and 60 roll the other. Line 70 prints the total. Here's a sample run:

```
I'M ROLLING A PAIR OF DICE
ONE OF THE DICE SAYS: 3
THE OTHER SAYS: 5
THE TOTAL IS 8
```

Here's another run:

```
I'M ROLLING A PAIR OF DICE
ONE OF THE DICE SAYS: 6
THE OTHER SAYS: 4
THE TOTAL IS 10
```

This program makes the computer flip a coin:

```
10 RANDOMIZE
20 R=RND(2)
30 IF R=1 THEN PRINT "HEADS": STOP
40 PRINT "TAILS"
```

Line 20 says R is a random number from 1 to 2; so R is either 1 or 2. If R is 1, line 30 makes the computer say HEADS; if R is 2, the computer skips line 30 and arrives at line 40, which prints TAILS instead. So the computer will print either HEADS or TAILS, depending on whether R is 1 or 2. Until you type RUN, you won't know which way the coin will flip; the choice is random. Each time you type RUN, the computer will flip the coin again; each time, the outcome is unpredictable.

Let's make the computer automatically re-run that program again and again, so that the computer automatically flips coins again and again, forever. To do that, put a loop in the program, by saying GO TO:

```
          10 RANDOMIZE
          20 R=RND(2)
Revised line→30 IF R=1 THEN PRINT "HEADS ";: GO TO 20
Revised line→40 PRINT "TAILS ";: GO TO 20
```

The computer will print something like this:

```
HEADS TAILS HEADS HEADS TAILS TAILS HEADS TAILS TAILS TAILS HEADS TAILS HEADS etc.
```

This program makes the computer talk about your friends:

```
10 RANDOMIZE
20 INPUT "TYPE THE NAME OF SOMEONE YOU LOVE...";N$
30 R=RND(3)
40 IF R=1 THEN PRINT N$;" HATES YOUR GUTS": GO TO 20
50 PRINT N$;" LOVES YOU, TOO": GO TO 20
```

Line 20 makes the computer ask:

```
TYPE THE NAME OF SOMEONE YOU LOVE...?
```

Suppose the human says SUZY. Then N$ is SUZY. Line 30 says R is a random number from 1 to 3. If R is 1, line 40 makes the computer say:

```
SUZY HATES YOUR GUTS
```

If R is 2 or 3, the computer arrives at line 50, which prints:

```
SUZY LOVES YOU, TOO
```

In that program, the chance is only 1 out of 3 that the computer will say "HATES YOUR GUTS". The chance is 2 out of 3 that the computer will be nicer, and say "LOVES YOU, TOO". Here's a sample run:

```
RUN
TYPE THE NAME OF SOMEONE YOU LOVE...? SUZY
SUZY LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? JOAN
JOAN HATES YOUR GUTS
TYPE THE NAME OF SOMEONE YOU LOVE...? ALICE
ALICE LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? FRED
FRED LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? UNCLE CHARLIE
UNCLE CHARLIE HATES YOUR GUTS
etc.
```

This program predicts what will happen to you today:

```
10 RANDOMIZE
20 PRINT "YOU WILL HAVE A ";
30 R=RND(5)
40 IF R=1 THEN PRINT "WONDERFUL";
50 IF R=2 THEN PRINT "BETTER-THAN-AVERAGE";
60 IF R=3 THEN PRINT "SO-SO";
70 IF R=4 THEN PRINT "WORSE-THAN-AVERAGE";
80 IF R=5 THEN PRINT "TERRIBLE";
90 PRINT " DAY TODAY"
```

The computer will say—

```
YOU WILL HAVE A WONDERFUL DAY TODAY
```

or—

```
YOU WILL HAVE A TERRIBLE DAY TODAY
```

or some in-between comment. For inspiration, run that program when you get up in the morning.

You've seen that RND(5) is a random number from 1 to 5: it's 1 or 2 or 3 or 4 or 5. To get a random *decimal* between 0 and 1, say just RND instead of RND(5).

RND makes the computer produce a decimal that's bigger than 0 but less than 1. For example, the decimal might be .284591.

PIXELS

Your computer's screen is divided into thousands of little rectangles, called *pixels*. The pixel in the upper-left corner of the screen is called pixel 0,0; just to the right of it is pixel 1,0; then comes pixel 2,0; etc. Underneath pixel 0,0 is pixel 0,1. Here are the positions of the pixels:

pixel 0,0	*pixel 1,0*	*pixel 2,0*	*pixel 3,0*	*pixel 4,0*	*pixel 5,0*	*etc.*
pixel 0,1	*pixel 1,1*	*pixel 2,1*	*pixel 3,1*	*pixel 4,1*	*pixel 5,1*	*etc.*
pixel 0,2	*pixel 1,2*	*pixel 2,2*	*pixel 3,2*	*pixel 4,2*	*pixel 5,2*	*etc.*
pixel 0,3	*pixel 1,3*	*pixel 2,3*	*pixel 3,3*	*pixel 4,3*	*pixel 5,3*	*etc.*

Each pixel's name consists of two numbers. The first number is called the *X coordinate*; the second number is called the *Y coordinate*. For example, if you're talking about pixel 5,3, its X coordinate is 5, and its Y coordinate is 3. The X coordinate tells how far to the right the pixel is; the Y coordinate tells how far down; so the pixel 5,3 is the pixel that's 5 to the right and 3 down.

So on the computer, the Y coordinate measures how far *down*. If you had the misfortune to read an old-fashioned math textbook (in which the Y coordinate measured how far *up*), you'll have to reverse your thinking!

How many pixels are on the screen? If you're using *high resolution*, the screen contains *many* pixels; each pixel is tiny. If you're using *low resolution*, the screen contains *few* pixels; each pixel is huge. You can choose either low resolution or high resolution.

You can make a pixel become white, or black, or—if you're lucky—a more exotic color. By writing a program that changes the colors of the pixels, you can create luscious art.

To find out how to make *your* computer change the colors of the pixels, read the appendix.

MUSIC

If your computer's fancy, it can play music. For details, read the appendix.

LOOP TECHNIQUES

INCREMENTING

Here's a strange program:

```
10 A=5
20 A=3+A
30 PRINT A
```

Line 20 means: the new A is 3 plus the old A. So the new A is 3+5, which is 8. Line 30 prints:

```
 8
```

In that program, line 10 puts 5 into box A:

```
10 A=5
20 A=3+A
30 PRINT A
```

Line 10 puts 5 into box A:

Box A [5]

When the computer sees line 20, it examines the right side of the equation, and sees the 3+A. Since A is 5, the 3+A is 3+5, which is 8. So line 20 says: A=8. The computer puts 8 into box A:

Box A [8]

Line 30 prints 8.

Here's another weirdo:

```
10 B=6
20 B=B+1
30 PRINT B*2
```

Line 20 says the new B is "the old B plus 1". So the new B is 6+1, which is 7. Line 30 prints:

```
 14
```

In that program, line 10 says B is 6; but line 20 increases B, by adding 1 to B; so B becomes 7. Programmers say that B has been *increased* or *incremented*. In line 20, the "1" is called the *increase* or the *increment*.

DECREMENTING

The opposite of *increment* is *decrement*:

```
10 J=500
20 J=J-1
30 PRINT J
```

Line 10 says J starts at 500; but line 20 says the new J is "the old J minus 1"; so the new J is 500-1, which is 499. Line 30 prints:

```
 499
```

In that program, J was *decreased* (or *decremented*). In line 20, the "1" is called the *decrease* (or *decrement*).

COUNTING

Suppose you want the computer to count, starting at 3, like this:

```
 3
 4
 5
 6
 7
 8
etc.
```

This program does it, by a special technique:

```
10 C=3
20 PRINT C
30 C=C+1
40 GO TO 20
```

In that program, C is called the *counter*, because it helps the computer count. Line 10 says C starts at 3. Line 20 makes the computer print C; so the computer prints:

```
 3
```

Line 30 increases C, by adding 1 to it; so C becomes 4. Line 40 sends the computer back to line 20, which prints the new value of C:

```
 4
```

Then the computer comes to line 30 again, which increases C again; so C becomes 5. Line 40 sends the computer back to line 20 again, which prints:

```
 5
```

The program is an infinite loop: the computer will print 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and so on, forever, unless you give an abortion.

Here's the general procedure for making the computer count:

1. Start C at some value (such as 3).
2. Use C. (For example, tell the computer to PRINT C.)
3. Increase C (by saying C=C+1).
4. GO back TO step 2.

To read the printing more easily, put a semicolon at the end of the PRINT statement:

```
  10 C=3
→→20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The semicolon makes the computer print horizontally:

```
 3  4  5  6  7  8  etc.
```

This program makes the computer count, starting at 1:

```
→→10 C=1
  20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The computer will print 1, 2, 3, 4, etc.

This program makes the computer count, starting at 0:

```
→→10 C=0
  20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The computer will print 0, 1, 2, 3, 4, etc.

Let's make the computer print a table, showing each number and its square, like this:

```
NUMBER          SQUARE
 0               0
 1               1
 2               4
 3               9
 4               16
 5               25
 6               36
 7               49
 8               64
 etc.
```

This program does it:

```
→→5 PRINT "NUMBER","SQUARE"
  10 C=0
→→20 PRINT C,C*C
  30 C=C+1
  40 GO TO 20
```

Line 5 prints the headings. Lines 10-40 make the computer count (0, 1, 2, 3, etc.); but instead of printing just C, line 20 also prints C*C.

Let's make the computer print a table that shows the squares of decimals, like this:

```
NUMBER          SQUARE
 0               0
 .1              .01
 .2              .04
 .3              .09
 .4              .16
 .5              .25
 .6              .36
 .7              .49
 .8              .64
 .9              .81
 1               1
 1.1             1.21
 1.2             1.44
 1.3             1.69
 1.4             1.96
 1.5             2.25
 etc.
```

To make the computer do that, just change line 30, so that the computer increases by .1 instead of by 1:

```
  5 PRINT "NUMBER","SQUARE"
  10 C=0
  20 PRINT C,C*C
→→30 C=C+.1
  40 GO TO 20
```

Let's make the computer give this quiz:

What's the capital of Nevada?
What's the chemical symbol for iron?
What word means 'brother or sister'?
What was Beethoven's first name?
How many cups are in a quart?

To make the computer score the quiz, we'll have to tell it the correct answers, which are:

Carson City
Fe
sibling
Ludwig
4

So our program will contain this data:

```
10 DATA WHAT'S THE CAPITAL OF NEVADA,CARSON CITY
20 DATA WHAT'S THE CHEMICAL SYMBOL FOR IRON,FE
30 DATA WHAT WORD MEANS 'BROTHER OR SISTER',SIBLING
40 DATA WHAT WAS BEETHOVEN'S FIRST NAME,LUDWIG
50 DATA HOW MANY CUPS ARE IN A QUART,4
```

Tell the computer to READ the data:

```
100 READ Q$,A$
```

That line reads a pair of data: it reads a question (Q$) and the correct answer (A$). The next step is to make the computer ask the question, and wait for the human's response:

```
110 PRINT Q$;
120 INPUT "??";H$
```

Line 110 prints the question. Line 120 prints question marks after the question, and waits for the human to respond; the human's response is called H$.

To complete the program, evaluate the human's response. If the human's response (H$) is the correct answer (A$), make the computer say "CORRECT", and then GO TO the next question:

```
130 IF H$=A$ THEN PRINT "CORRECT": GO TO 100
```

But if the human's response is wrong, make the computer say "NO" and reveal the correct answer:

```
140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

Here's a sample run:

```
RUN
WHAT'S THE CAPITAL OF NEVADA??? LAS VEGAS
NO, THE ANSWER IS: CARSON CITY
WHAT'S THE CHEMICAL SYMBOL FOR IRON??? FE
CORRECT
WHAT WORD MEANS 'BROTHER OR SISTER'??? I GIVE UP
NO, THE ANSWER IS: SIBLING
WHAT WAS BEETHOVEN'S FIRST NAME??? LUDVIG
NO, THE ANSWER IS: LUDWIG
HOW MANY CUPS ARE IN A QUART??? 4
CORRECT
OUT OF DATA
```

To give a quiz about different topics, change the data in lines 10-50.

Instead of making the computer say OUT OF DATA, let's make it say:

```
I HOPE YOU ENJOYED THE QUIZ
```

To do that, write an end mark and an end routine:

```
60 DATA END,END
100 READ Q$,A$: IF Q$="END" THEN PRINT "I HOPE YOU ENJOYED THE QUIZ": END
```

Let's make the computer count how many questions the human answered correctly. To do that, we need a counter. As usual, we'll call it C:

```
→→5 C=0
  10 DATA WHAT'S THE CAPITAL OF NEVADA,CARSON CITY
  20 DATA WHAT'S THE CHEMICAL SYMBOL FOR IRON,FE
  30 DATA WHAT WORD MEANS 'BROTHER OR SISTER,SIBLING
  40 DATA WHAT WAS BEETHOVEN'S FIRST NAME,LUDWIG
  50 DATA HOW MANY CUPS ARE IN A QUART,4
  60 DATA END,END
→→100 READ Q$,A$: IF Q$="END" THEN PRINT "YOU ANSWERED";C;"OF THE QUESTIONS CORRECT
  LY": PRINT "I HOPE YOU ENJOYED THE QUIZ": END
  110 PRINT Q$;
  120 INPUT "??";H$
→→130 IF H$=A$ THEN PRINT "CORRECT": C=C+1: GO TO 100
  140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

Line 5 begins the counter at 0 (because at the beginning, the human hasn't answered any questions correctly yet). When the program ends, line 100 prints the value of the counter. Line 130 makes sure that each time the human answers a question correctly, the counter increases.

Line 100 prints a message such as:

```
YOU ANSWERED 2 OF THE QUESTIONS CORRECTLY
```

It would be nicer to print—

```
YOU ANSWERED 2 OF THE 5 QUESTIONS CORRECTLY
YOUR SCORE IS 40 %
```

— or, if the quiz were changed to include 8 questions:

```
YOU ANSWERED 2 OF THE 8 QUESTIONS CORRECTLY
YOUR SCORE IS 25 %
```

To make the computer print the 5 or the 8, we must make the computer count how many questions were asked. So we need another counter. Since we already used C to count the number of correct answers, we'll use D to count the number of questions asked. Like C, D must start at 0; and we must increase D, by adding 1 each time another question is asked:

```
→→5 C=0: D=0
  10 DATA WHAT'S THE CAPITAL OF NEVADA,CARSON CITY
  20 DATA WHAT'S THE CHEMICAL SYMBOL FOR IRON,FE
  30 DATA WHAT WORD MEANS 'BROTHER OR SISTER',SIBLING
  40 DATA WHAT WAS BEETHOVEN'S FIRST NAME,LUDWIG
  50 DATA HOW MANY CUPS ARE IN A QUART,4
  60 DATA END,END
→→100 READ Q$,A$: IF Q$="END" THEN PRINT "YOU ANSWERED";C;"OF THE";D;"QUESTIONS COR
  RECTLY": PRINT "YOUR SCORE IS";C/D*100;"%": PRINT "I HOPE YOU ENJOYED THE QUIZ":
  END
  110 PRINT Q$;
→→115 D=D+1
  120 INPUT "??";H$
  130 IF H$=A$ THEN PRINT "CORRECT": C=C+1: GO TO 100
  140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

SUMMING

Let's make the computer imitate an adding machine, so a run looks like this:

```
RUN
NOW THE SUM IS 0
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 5
NOW THE SUM IS 5
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 3
NOW THE SUM IS 8
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 6.1
NOW THE SUM IS 14.1
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM -10
NOW THE SUM IS 4.1
etc.
```

Here's the program:

```
10 S=0
20 PRINT "NOW THE SUM IS";S
30 INPUT "WHAT NUMBER DO YOU WANT TO ADD TO THE SUM";X
40 S=S+X
50 GO TO 20
```

Line 10 starts the sum at 0. Line 20 prints the sum. Line 30 asks the human what number to add to the sum; the human's number is called X. Line 40 adds X to the sum, so the sum changes. Line 50 makes the computer go to line 20, which prints the new sum. Lines 20-50 form an infinite loop, which you must abort.

Here's the general procedure for making the computer find a sum:

1. Start S at 0.
2. Use S. (For example, tell the computer to PRINT S.)
3. Find out what number to add to S. (For example, tell the human to input a number X.)
4. Increase S (by saying S = S + the number to be added).
5. GO back TO step 2.

CHECKING ACCOUNT

If your bank's nasty, it treats your checking account like this: you must pay a 10¢ service charge for each good check; you must pay a $5 penalty for each check that bounces; and the money you've deposited earns no interest.

This program makes the computer imitate such a bank:

Start the sum at 0→→→
```
10 S=0
```
Chat with the human→→
```
20 PRINT "YOUR CHECKING ACCOUNT CONTAINS";S
30 INPUT "DEPOSIT OR WITHDRAW";A$
40 IF A$="DEPOSIT" THEN GO TO 100
50 IF A$="WITHDRAW" THEN GO TO 200
60 PRINT "PLEASE SAY DEPOSIT OR WITHDRAW": GO TO 30
```
Deposit some money→→→
```
100 INPUT "HOW MUCH DO YOU WANT TO DEPOSIT";D
110 S=S+D
120 GO TO 20
```
Withdraw some money→→
```
200 INPUT "HOW MUCH DO YOU WANT TO WITHDRAW";W
210 W=W+.10
220 IF W<=S THEN PRINT "OKAY": S=S-W: GO TO 20
230 PRINT "THAT CHECK BOUNCED": S=S-5: GO TO 20
```

Line 10 starts the sum at 0. Line 20 prints the sum. Line 30 asks whether the human wants to deposit or withdraw.

What happens if the human says DEPOSIT? The computer goes from line 40 to line 100, which asks how much to deposit. Line 110 increases the sum in the account, by adding the deposit. Line 120 sends the computer back to line 20, which tells the human the new sum (i.e., the balance), and gets ready for the next transaction.

What happens if the human says WITHDRAW instead of deposit? The computer goes from line 50 to line 200, which asks how much to withdraw. Line 210 adds the 10¢ charge to the withdrawal amount. Line 220 compares the withdrawal amount (W) against the sum in the account (S). If W<=S, the computer says OKAY, processes the check, decreases the sum in the account (by subtracting W), and gets ready for the next transaction (by going back to line 20). If W>S instead, the computer says THAT CHECK BOUNCED, decreases the sum in the account by $5 (the penalty fee), and goes back to line 20, for the next transaction.

That program will annoy some customers. For example, suppose you have $1 in your account, and you try to write a check for 95¢. Since 95¢ + the 10¢ service charge = $1.05, which is more than you have in your account, your check will bounce, and you'll be penalized $5, so your balance will become *negative* $4, and the bank will demand that *you* pay the *bank* $4— just because you wrote a check for 95¢!

Another nuisance is when you leave town permanently, and want to close your account. If your account contains $1, you cannot get the dollar back! The most you can withdraw is 90¢, because 90¢ + the 10¢ service charge = $1.

In those cases, the program is "nasty", and makes the customers hate the bank— and hate the computer!

The bank should change the program, to make it act friendlier. Here's how. It should insert this line:

```
15 IF S<0 THEN S=0
```

It should eliminate line 210. It should retype lines 220-230, as follows:

```
220 IF W<=S THEN PRINT "OKAY": S=S-W-.10: GO TO 15
230 PRINT "THAT CHECK BOUNCED": S=S-5: GO TO 15
```

Here's why. Line 15 is kind, because it makes the computer change a negative sum to 0, so that the bank never accuses the customer of owing money. Eliminating line 210 is kind, because now the computer ignores the 10¢ service charge when comparing W against S in line 220. The new versions of lines 220 and 230 make sure that the computer goes to line 15 (the kind line) before arriving at line 20.

But some banks would argue that the new program is *too* kind! For example, if a person has only 1¢ in his account, and writes a check for a million dollars, which bounces, the new program charges him only 1¢ for the bad check; $5 might be more reasonable.

Moral: the hardest thing about programming is choosing your goal— deciding what you *want* the computer to do.

BETTING

This program lets you bet:

```
10 RANDOMIZE
20 S=100
30 PRINT "YOU HAVE";S;"DOLLARS."

40 INPUT "HOW MANY DOLLARS DO YOU WANT TO BET";B
50 IF B>S THEN PRINT "YOU DON'T HAVE THAT MUCH MONEY!  YOU MUST BET LESS.": GO TO 40
60 IF B<0 THEN PRINT "YOU CAN'T BET LESS THAN NOTHING!": GO TO 40
70 IF B=0 THEN PRINT "I GUESS YOU DON'T WANT TO BET ANYMORE.": GO TO 300

80 INPUT "DO YOU WANT TO BET ON HEADS OR TAILS";A$
90 IF A$="HEADS" THEN GO TO 200
100 IF A$="TAILS" THEN GO TO 200
110 PRINT "PLEASE SAY 'HEADS' OR 'TAILS'": GO TO 80

200 R=RND(2)
210 IF R=1 THEN R$="HEADS"
220 IF R=2 THEN R$="TAILS"
230 PRINT "THE COIN SAYS ";R$;"."

240 IF R$=A$ THEN PRINT "YOU WIN";B;"DOLLARS": S=S+B: GO TO 30
250 PRINT "YOU LOSE";B;"DOLLARS": S=S-B: IF S>0 THEN GO TO 30

260 PRINT "YOU'RE BROKE!  TOO BAD!"
300 PRINT "THANK YOU FOR PLAYING WITH ME.  YOU WERE FUN TO PLAY WITH."
310 PRINT "I HOPE YOU PLAY AGAIN SOMETIME."
```

Here's how the program works. Lines 20 and 30 make the computer say:

```
YOU HAVE 100 DOLLARS.
```

Line 40 makes the computer ask:

```
HOW MANY DOLLARS DO YOU WANT TO BET?
```

Lines 50-70 make sure the bet is reasonable. Line 80 makes the computer ask:

```
DO YOU WANT TO BET ON HEADS OR TAILS?
```

Lines 90-110 make sure the human says HEADS or TAILS.

Lines 200-230 make the computer flip a coin. Lines 240-250 determine whether the human won or lost the bet, and then send the computer back to line 30, for another round (if the human isn't broke yet). Lines 260-310 say good-bye to the human.

Here's a sample run:

```
YOU HAVE 100 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 120
YOU DON'T HAVE THAT MUCH MONEY!  YOU MUST BET LESS.
HOW MANY DOLLARS DO YOU WANT TO BET? 75
DO YOU WANT TO BET ON HEADS OR TAILS? HEADS
THE COIN SAYS TAILS.
YOU LOSE 75 DOLLARS.
YOU HAVE 25 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 10
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS TAILS.
YOU WIN 10 DOLLARS.
YOU HAVE 35 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 35
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS HEADS.
YOU LOSE 35 DOLLARS.
YOU'RE BROKE!  TOO BAD!
THANK YOU FOR PLAYING WITH ME.  YOU WERE FUN TO PLAY WITH.
I HOPE YOU PLAY AGAIN SOMETIME.
```

To make the output prettier, insert these lines:

```
30 PRINT
31 PRINT "YOU HAVE";S;"DOLLARS.  HERE THEY ARE:"
32 FOR I = 1 TO S
33     PRINT "$";
34 NEXT
35 PRINT
```

Now the run looks like this:

```
RUN

YOU HAVE 100 DOLLARS.  HERE THEY ARE:
$$$
```

SERIES

Let's make the computer add together all the numbers from 7 to 100. In other words, let's make the computer find the sum of this series: 7+8+9+...+100. Here's how:

```
Start the Sum at 0→→→→→→→→→→→→→→→→→→→→→→→10 S=0
Make I go from 7 to 100→→→→→→→→→→→→→→→→→→20 FOR I = 7 TO 100
Increase the Sum, by adding each I to it→→30     S=S+I
                                          40 NEXT
Print the final Sum (which is 5029)→→→→→→→50 PRINT S
```

Let's make the computer add together the *squares* of all the numbers from 7 to 100. In other words, let's make the computer find the sum of this series: (7 squared) + (8 squared) + (9 squared) + ... + (100 squared). Here's how:

```
          10 S=0
          20 FOR I = 7 TO 100
New line→→30     S=S+I*I
          40 NEXT
          50 PRINT S
```

It's the same as the previous program, except that line 30 says to add I*I instead of I. Line 50 prints the final sum, which is 338259.

DATA SUMS

This program adds together the numbers in the data:

```
10 S=0
20 DATA 5, 3, 6.1, etc.
30 DATA etc.
40 DATA etc.
50 DATA 0
60 READ X: IF X=0 THEN PRINT S: END
70 S=S+X
80 GO TO 60
```

Line 10 starts the sum at zero. Lines 20-40 contain the numbers to be added. The zero in line 50 is an end mark. Line 60 reads an X from the data. If the X it reads is zero, the end of the data has been reached, so we want the computer to print the sum (S) and stop; but if the X it reads is *not* zero, the computer proceeds from line 60 to line 70, adds X to the sum, and goes from line 80 to line 60, which reads another X.

CHAPTER 5

AMAZING APPLICATIONS

CHANGE YOUR LIFE

MAGIC SIX

Computers are changing our lives in many ways. Today, computer applications fall into six broad categories....

games: computers can be even more fun than sex!
real-time control: you can make the computer secretly control other machines!
communication: you need a computer, to communicate with other people quickly and easily
accounting: the computer can tell you how to get rich, by increasing your profits
information retrieval: the computer can store, in its data banks, anything you wish!
education/research: the computer can tutor you, to make you think at "lightning" speed

Let's look at them more closely....

GAMES

In the old days, kids went to the "pool hall", to spend money on playing pool— and even more money on betting about who would win. When pinball machines came, kids switched from pool to pinball, and went to "pinball arcades" to pop quarters into pinball machines.

Today, pinball machines are being replaced by computerized video games— and business is booming. Many teenagers are getting hooked on those computer games, which are even more addictive than "drugs, sex, and rock-and-roll". Some teenagers spend $10 per day, to feed their lust for computer games.

According to *Newsweek Magazine*, more money is being spent on computerized games now than on movies or records. In other words, the computer-game industry is bigger than the entire movie industry, and bigger than the entire record industry. So even though movie stars and rock-music stars can get rich, people who create computer games can get even richer.

Parents complain that teenagers waste too much money and time playing those games, and that too many of the games glorify violence: the object is to kill the mean invaders from outer space. On the other hand, by playing the games you improve your hand-eye coordination, improve your ability to concentrate and stay alert, and learn how to deal with computers.

And anyway, aren't computer games better to spend money on than "drugs, sex, and rock-and-roll"? And isn't it better to "take out your aggressions" on space invaders than on people?

Some arcades are starting to post signs, prohibiting drinking and smoking and drugs, and barring teenagers during school hours. So if your child frequents such an arcade, you know your child is in a good, safe place.

To avoid spending so much money on arcade games, buy a personal computer for your home, and buy games to run on your computer. Although the programs are expensive, they're cheaper than popping in so many quarters. And maybe you can start charging your neighbors!

Not all games are violent. For your home computer, you can buy games that quiz your ability to spell and your knowledge of history.

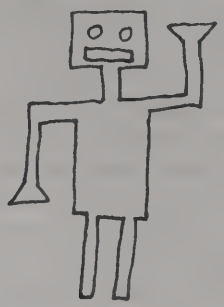
Some games throw you in the middle of a real-life situation, and give you practice in coping. For example, some games begin by saying that "you are President of the United States", or "you are running the control room of the Three-Mile Island nuclear power plant", or "you are the game warden for New Jersey", or "you are an air-traffic controller", or "you are a woman deciding which birth-control method to use". You try to make the right decisions— and the computer shows you the consequences. If you make a wrong decision, don't worry: it's only a game. You learn from the experience of playing with the computer.

REAL-TIME CONTROL

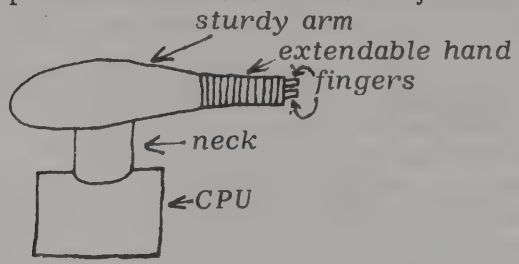
Today, computers hide inside many devices. You can find computers inside digital watches, digital clocks, microwave ovens (to control the timer), and cars (to reduce the pollution).

You can also find computers inside robots. Yes, robots exist. You can buy four kinds of robots....

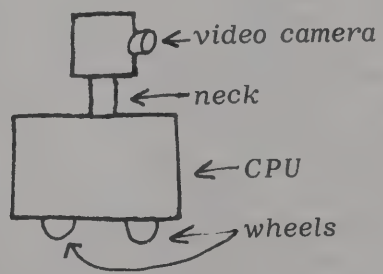
an *entertainment robot* looks like a person— it walks and talks— and is just for fun:



an *industrial robot* is a sturdy computerized arm that moves objects in factories:



an *experimental robot* is a robot that tries to act as smart as a person:



an *educational robot* is a low-cost blob you can easily program to move:

In Japan, if you walk into an automobile factory at night, you'll see that all the cars are built by industrial robots: in the entire factory, there's only one human, who just watches. Robots are already walking on the bottom of the ocean, to hunt for oil. Industry and goverment are requesting robots for work in hostile environments, such as coal mines and outer space.

Today, large computers are controlling rockets, missiles, space ships, and nuclear bombs.

My favorite example of computer control is the Casio VL-1, which lists for $50. That price includes a piano-style keyboard, a CPU, a ROM (that can make the machine imitate many kinds of musical instruments), a RAM (that memorizes whatever you play), a speaker (so you can hear the music), and a screen (that shows the notes, and also prints the answer to any math questions you wish— because the musical instrument is also a calculator). All that for just $50! It makes an excellent present for Christmas, a birthday, or graduation: give it to your favorite future rock-and-roll star.

COMMUNICATION

In prehistoric times, the only way people could communicate was by using their mouths and hands. Then came instruments for writing (the pen, printing press, and typewriter), hearing (the telephone, phonograph, tape recorder, and radio), and seeing (movies and television). Finally, we have the ultimate communication device: the computer.

A computerized typewriter is called a *word processor*: it helps edit your writing. The fanciest word processors do all these tasks *automatically*:

press the carriage return
keep the right margin perfectly straight
center the headings
move paragraphs apart, so that you can insert a new paragraph between them
correct your spelling (and find typographical errors)
hyphenate, by looking up the words in the computer's dictionary
number the pages
make room for footnotes
create a table of contents
create an alphabetical index, showing the page where you used each important word
store the final document on the disk
make the printer print multiple copies of the document
personalize the copies, by sending each one to a different name and address

The newest word processors are super-easy to learn: you can learn how to use them in a few minutes, and become an "experienced expert" in less than a day. Because word processing is so quick and easy— and so much fun— executives are starting to do their own typing, instead of hiring secretaries.

Word processors are getting cheaper. Now, for under $600, companies such as Commodore and Coleco are selling complete word-processing systems. The $600 price includes everything: the program, CPU, RAM, ROM, keyboard, screen, permanent storage, and even the printer! Other word-processing systems are more expensive— the "typical" price is $3500— but the price is dropping rapidly. And if you already have a computer and a printer, you can turn your system into a word processor cheaply: just add a word-processing program, which costs anywhere from $40 to $600.

Because word processing is so easy, school kids are starting to use it, to type their homework papers. As a result, many little kiddies are learning how to type. Someday, maybe pens will become obsolete.

Even artists are starting to use computers. To create amazing art, attach your Apple to a video camera, a joystick, and paddles, and buy fancy "graphics" programs that let you draw pictures on your T.V. screen easily. It's as easy as taking a photograph— and as easy as doodling on a pad of paper. But the computerized results are much more interesting to look at.

Handicapped people— who can't see or hear or talk— are using computers to replace their eyes, ears, and vocal chords. The most amazing computerized eye is the Kurtzweil reading machine. It looks like a photocopy machine: you lay a book on it; but instead of making a copy of the book for you, it *reads* the book, by using its own camera and computer and voice synthesizer.

Many businesses use an *electronic mail system*. It's a computer that has lots of terminals and disks. You type a message on your terminal's keyboard; then you tell the computer to copy that message onto a disk, and tell the computer who to send the message to: that person is called the *addressee*. When the addressee walks up to his terminal, the computer tells him a message is waiting on disk for him. Then the addressee tells the computer to read him the mail you sent.

It's faster than waiting for a human mailman to deliver the mail. And it's faster than repeatedly trying to use a telephone that gives a "busy signal" or "no answer" or a getting past a secretary who says the person is "out of the office" or "at a meeting". The computer simply stores the message on disk until the addressee shows up, then transmits the message to him instantaneously.

The addressee can be thousands of miles away from you. The only restriction is that his terminal must be communicating with the same computer network as yours.

Instead of sending your message to one person, the computer can send copies of your message to *several* people, if you prefer. And it will tell you which of those people have read the message.

You can type your message on your terminal's keyboard— or you can use your voice! Bell Telephone has invented an easy way to store your voice on a disk, and wants the government to approve the method. But the answering-machine companies are warning the government that such approval would further strengthen Bell Telephone's monopoly.

Today, many people use *public information utilities*, which are big computers available to the general public at low cost. The most popular public information utilities are *The Source* and *The Compuserve Micronet*. The Source is owned by Reader's Digest Magazine; The Compuserve Micronet is owned by the same company that owns H&R Block Tax Consultants. To use one of those public information utilities, you need just three things: a computer terminal (or a personal computer), a modem (which attaches to a telephone), and a charge card (such as Visa). In the privacy of your own home, you attach your terminal (or personal computer) to a modem, which you attach to your telephone; then you dial the phone number for The Source or The Compuserve Micronet. The cost of using the service is low (typically $5 per hour), which is billed to your charge card. If you live in a major city, the $5-per-hour charge *includes* the cost of the phone call! From your own home, you can use the full powers of those giant computers— including such services as electronic mail, airline schedules, lists of good restaurants, and— most important— a data bank that contains all the important information that appeared in major newspapers during the past several years. That data bank is a computerized encyclopedia of all current events.

If you don't want to deal with those big computers, choose a smaller computer instead, such as a *computer bulletin board system (CBBS)*, which is a computer you can use for free and leave messages on. To use a CBBS, just sit in your own home with your own terminal and dial the phone number of your neighborhood's CBBS. On the CBBS's disk, you can leave whatever message you wish, for your neighbors to read. Some of the CBBS's are sexual: use them to find people to fall in love with. If you prefer phone a CBBS in another city, to get a computerized "pen pal". If you want to sell or buy a used computer, you can put a free ad on the CBBS's disk. Each CBBS is run by a friendly computer hobbyist who wants to be a "good guy" and let other people use his computer. If he's a programmer, he puts a message on the disk, saying he's selling his programs; he hopes you'll buy some of them, but you don't have to.

The French government plans to put a terminal in every French home. The terminal will replace the phone book. To "look up" a phone number, you type the person's name on the terminal. The government says that giving each home a terminal is cheaper than giving each home an updated phone book every year. And since you won't have to ever call "information", the phone company call fire the humans that are "information operators". So the French phone system will be computerized totally.

If you home contains a computer terminal, how can it get information from large computers that are far away? You have four choices....

use a phone

use your television's antenna: computer information transmitted via television is called "teletext"; it's becoming popular in France, England, Canada, and Japan

use "cable" television— which is especially exciting if the cable is two-way, so you can talk back!

use a satellite, up in the sky: American computer companies have gotten together and formed Satellite Business Systems

ACCOUNTING

Problem: you'd like to buy an Apple computer (because it's nifty, and plays good games, and you want the experience of having your own computer), but you can't afford to pay for it out of your own pocket. So you try to convince your boss that by giving you an Apple, his business will improve. Or— if you're your own boss— you try to convince your accountant that the Apple is a legitimate business expense, eligible for some sort of tax credit. How do you convince your boss or accountant that the Apple is a "serious and useful" machine, not just a toy?

During the past few years, many potential Apple customers pondered that question. And Apple's salesmen gave a simple, one-word solution: *Visicalc*.

Visicalc is a computer program that makes the Apple do business accounting easily. The program is easy to use and inexpensive. (It originally cost $99, though recently the price has risen to $250.) It enables the Apple to do, with ease, tasks that are awkward for human accountants— and awkward even for larger computers. In other words, it turns the Apple into a machine that's even more useful than a large IBM. Apple's television advertisement showed a television star— Dick Cavett— praising the Apple's ability to handle Visicalc.

Originally, the only computer that had Visicalc was the Apple. Today, you can get Visicalc on many other microcomputers: Radio Shack, Commodore, Atari, Hewlett-Packard, and the IBM Personal Computer. An imitation called *Supercalc* runs on all the other popular microcomputers (the microcomputers that use the CP/M operating system). For some computers, you can also get a souped-up version (called *Multiplan*) and an even fancier version (called *1-2-3*).

Visicalc was important, because it convinced business executives that a microcomputer can be a serious tool, and not just a toy. It led to a chain of events that forced IBM, Xerox, and other major companies to enter the microcomputer market.

Visicalc is a general accounting program that can be bent to serve any purpose. But if you want to solve a *particular* accounting problem— such as payroll, inventory, general ledger, accounts receivable, or accounts payable— you might prefer a particular accounting program instead of Visicalc. Particular accounting programs for microcomputers are expensive— they typically cost several hundred dollars— but nevertheless are far cheaper than on larger computers, where those programs cost several *thousand* dollars.

INFORMATION RETRIEVAL

On the computer's disks, you can store lots of data, about any topic you like: your friends, your customers, your clients, answers to a quiz, results of a survey, a catalog of all your books and stereo records and tapes and magazine clippings, a calendar of birthdays, shopping lists, memos, etc. But to store that data on a disk, you need a program. Writing a program to handle "data files on disks" is hard; so you'll probably want to *buy* a program instead. But you'll have a hard time finding a program that uses your favorite way to organize the data.

There's a new solution to the problem. It's called *PFS*, because it's a "personal filing system" that's a "powerful, flexible solution" and "pretty fucking simple" to use. It turns data files into a pleasure instead of a pain. It's inexpensive and runs on your Apple or IBM Personal Computer. Get it!

If your needs are very elaborate, PFS might not be fancy enough. But it's a good place to begin.

"Information retrieval" can lead to a problem, called *invasion of privacy*. If the government stores lots of information about you in its giant computers, is your privacy being invaded? Lawyers, the American Civil Liberties Union, Ralph Nader, the FBI, the CIA, the IRS, and the rest of the gang all have fun, debating whether government data banks should be abolished.

The computer can teach you many subjects, by "tutoring" you or "drilling" you or "playing educational games". But the most important skill you'll learn from the computer is: how to program! Then you can get a job as a programmer or consultant, and grab lots of bucks.

Computer programming is being taught in many colleges, high schools, and even elementary schools. Many colleges require that you pass a course in "computer programming" before you graduate. And so do some high schools!

Why are kids forced to learn how to "prove geometry theorems"? Answer: "because geometry helps train the mind". But computer programming does an even *better* job of training the mind. Once you learn how to use programming concepts, such as "IF...THEN" and "GO TO", your entire method of thinking becomes more logical, and you approach *all* problems in life more logically— with a greater awareness of the importance of understanding life's alternatives.

Because computer programming is so new and so easy, kids learn it quickly and soon outsmart their teachers, who then become scared of the kids. Instead of getting scared, the teacher should encourage those bright kids to help their classmates.

Computer programming is particularly worthwhile for teenagers. It's the only activity a teenager can excel in, without needing pats on the back from peers or teachers or other adults. As a competent computer programmer, you get an amazing sense of independence.

A philosopher has recommended that the government give each elderly citizen a microcomputer, free— because a microcomputer gives pleasant companionship to the lonely, and because it poses an intellectual challenge that helps sharpen the mind and prevent senility. That philosopher was joking— but only partly.

To teach an idiot a difficult task (such as "how to write subtle poetry"), you must understand that task thoroughly. The ultimate proof that you understand a task thoroughly is to teach it to the ultimate idiot, which is the computer. So the ultimate proof of your intelligence is your ability to write a program. In fact, computer programming is the highest form of human accomplishment— if the subject being programmed is sufficiently difficult and "non-computery"— such as writing subtle poetry.

Unfortunately, "writing subtle poetry" hasn't been successfully programmed yet. We programmers still have a long way to go!

You have *natural intelligence* (although some people accuse you of having "natural stupidity" instead). Computers have *artificial intelligence*. If somebody is studying *the problem of artificial intelligence*, he's studying how to make the computer become as smart and as flexible as a human.

For example, a human can drive a car. Can a computer drive a car? No, because a computer isn't smart enough. For example, if the computer were driving a car, and the car reached a red light, and the computer were programmed to "stop on red", the computer would stop the car and keep it stopped as long as the light remained red— even if the light were broken and remained red forever. After waiting an hour at the red light, a human would deduce that the light was broken and would move ahead (cautiously); but a computer wouldn't be that smart— unless it were properly programmed. But how do you program a computer to handle all possible unexpected events— such as broken traffic lights, vandalized traffic signs, dogs dashing across the street, drivers who signal "left" when they mean "right", etc.? No programmer has solved that problem yet. Can you? Until you can solve such problems, the computer's "artificial intelligence" will remain inferior to our own natural common sense.

Too bad! We can send a rocket to the moon, but we can't even send a computerized car around the block.

I hope that someday my computer will become smart enough to write this book, so I can relax and go to sleep. If you write a program that writes computer books, I'll be glad to buy it.

Until that miraculous day, I remain here, trying to make myself (and my computers) more educated.

FALL IN LOVE

HISTORY

Can the computer help you fall in love? Here are some famous attempts, in chronological order. (I've rounded all dates to the nearest 5 years.)

TELEVISION LOVE (1960)

A computer appeared on national television, to make people fall in love.

Guys and gals in the audience answered questionnaires about their personality and fed them into the computer. The computer chose the guy and gal that were most compatible. That guy and gal had their first blind date on national television.

Each week, that scenario was repeated: the computer chose another couple from the audience.

Each lucky couple appeared on the show again several weeks later, so the audience could find out whether the couple was in love.

One of the couples was unhappy: the gal didn't like the guy, even though she *wanted* to like him. She volunteered to be hypnotized. So, on national television, a hypnotist made her fall in love with her partner.

The computer was a huge Univac. Today, the same kind of matching could be done with a microcomputer. Any volunteers?

COMPUTER-DATING SERVICES (1965)

College students began relying on computers, to find dates. Here's how the typical computer-dating service worked....

You answered a long questionnaire— about 8 pages. The questionnaire asked about your sex, age, height, weight, hair color, race, religion, how often you drank and smoked, how "handsome" or "attractive" you were (on a scale of 1 to 10), how far you wanted to go on your first date, whether you wanted to get married soon, and how many children you'd like. It also asked many questions about your personality.

One of the questions was:

Suppose you receive in the mail some spoons you didn't order. The accompanying note says the spoons were sent by a charitable organization, and begs you to either send a contribution or return the spoons. You don't like the spoons. What will you do?

1. *Keep the spoons without paying.*
2. *Return the spoons.*
3. *Pay for the spoons.*

Another question was:

A girl returned from her date after curfew. Her excuse was that her boyfriend's car broke down. What's your reaction?

Again, you had a multiple-choice answer. One of the choices was, "Ha!"

For each question, you had to say how *you* would answer it, and how you'd want your *date* to answer it. That was tough. What if you wanted your date to be stunningly beautiful but also humble? What if you wanted to meet somebody who was so ugly that the person would be "desperate" to have sex with anybody? Such issues were debated in college dorms throughout the nation.

After completing the questionnaire, you mailed it with about $10 to the computer-dating service. Within two months, the service would send you the names, addresses, and phone numbers of at least 5 people you could date. If your personality was very easy to match, the service might send you *more* than 5 names; but even if your personality was lousy, you'd get at least 5. Moreover, you'd get updates every few months, throughout the year. That way, you could be matched with people who enrolled several months after you.

The most popular computer-dating service was "Operation Match", started by students at Harvard. Its main competitor was "Contact", started by students at M.I.T. Both services quickly became profitable and had subscribers from all across the country.

One gal had such a wonderful personality that the computer matched her with 110 guys! She had to explain to her mom why 110 guys were always on the phone. And she had to figure out how to say "no" to 109 of them.

Two gals who shared a dorm room fed their names into the computer. One of the gals got matched to the other gal's boyfriend. They didn't stay roommates long.

When I was a freshman, I applied to *both* services, to make sure I'd meet "the gal of my dreams". "Contact" sent me names of gals at prestigious schools (such as Wellesley and Bennington), while "Operation Match" sent me names of gals at schools such as the State University of New York at Albany. I thought I was the only nut desperate enough to apply to *both* services. But I got a surprise: when I looked down the list of names from "Contact" and the list of names from "Operation Match", I found a gal who appeared on *both* lists! She, like me, had been desperate enough to apply to both services! And both computers agreed that she and I would be a perfect match!

I had a date with her. But I couldn't stand her.

When I had answered the questionnaire, I was a very bashful boy, so the computer matched me to bashful girls. But by the time I received the computer print-out, I'd become wilder, and the girls the computer matched me to were no longer "my type".

"Contact" raised its price to $15, then $20. But $20 was still cheap— when you consider what you were getting.

"Contact" put a cute ad in the newspaper. It seemed to be selling groceries. It said, "Dates— 2¢ per pound". The ad went on to explain that one gal got so many dates that, when she totaled the weight of their bodies, she figured they cost her 2¢ per pound.

THE DARTMOUTH DATER (1965)

At Dartmouth College, which was all-male, one of the students wrote a cruel program to evaluate dates. It asked lots of "practical" questions, such as "is she pretty" and "how far away does she live" and "does she have a car". I put down that I was dating a 14-year-old girl who was 7 feet tall and weighed 300 pounds but had a perfect personality: I gave her personality a 10, and even said that she lived nearby and had a car. In spite of her excellent personality, the computer didn't like her. The computer said:

```
SHE MUST BE PREGNANT. WHERE DID YOU GET THAT PIG?
WORST SCORE YET PRODUCED BY THIS COMPUTER!
```

VIDEO DATING (1975)

During the 1970's, people wanted everything to be natural. They wanted "natural food" and "natural love".

Computerized love seemed unnatural, and its popularity declined. "Operation Match" and "Contact" went out of business.

They were replaced by *video dating*. You go to a *video-dating service*, which makes videotapes of guys and gals and shows them on private television. When you apply to the service, the service makes a videotape of you. Then the service shows you videotapes of people of the opposite sex, and you choose the person you'd like to date. That way, you never have a "blind" date: you see the person on videotape before you make the date.

The video-dating service tapes *thousands* of people. But you don't have enough time to look at thousands of tapes. So the video-dating service tells you to answer a questionnaire; your questionnaire is fed into a computer, which tells which people you're most compatible with. Then you look at those people's tapes.

COMPUTER DANCING (1975)

At a Connecticut prep school (Hotchkiss), the head of the computer center arranged a "computer dance".

All the students in the school answered questionnaires, which were fed into a computer. The computer matched the boys with the girls, so that each boy got one girl. The boy had to take the girl to the dance.

The night before the computer announced the dancing partners, the computer-center staff stayed up late. In the morning, when the students came to school, the students found all the halls covered with strips of punched paper tape, saying, in billboard-style letters, messages such as "George Smith & Mary Jones". If you were a student, you looked up and down the halls (your heart beating quickly), to find the tape that had your name on it, with the name of your mysterious computer lover.

Shrieks and groans. "Aarrgghh! You wouldn't *believe* who the computer stuck me with!"

COMPUTER WEDDINGS (1980)

Here's how the first true "computer marriage" occurred....

One company's terminal was attached to another company's computer. A programmer at the first company often asked a programmer at the second company for help. They contacted each other by typing messages on their terminals, and let the computer relay the messages back and forth. One of the programmers was a guy, the other was a gal, and they fell in love, even though they had never met. Finally, the guy typed on his terminal, "Let's get married". The gal typed back, "Yes". And so they got engaged— even though they had never met. Their marriage ceremony used three terminals: one for the guy, one for the gal, and one for the mininster. The minister typed the questions at his own terminal; then the guy and gal typed back, "I do".

Another example of a computer wedding is "Reverend Apple". It's an Apple computer, programmed to perform a marriage ceremony. It performed its first marriage on Valentine's Day, 1981. The groom was a guy named Richard; the bride was a gal named Debbie. The computer printed the standard wedding-ritual text on the screen, and then asked the usual questions. Insead of answering "I do", the bride and groom just had to type "Y".

"Reverend Apple" is smart. For example, if the bride or groom types "N" instead of "Y", the computer beeps, tells the couple to try again, and repeats the question.

The program was written by M.E. Cavanaugh at the request of Rev. Jon Jaenisch, who stood by "Reverend Apple" while the ceremony was being performed. Rev. Jaenisch is a minister of the Universal Life Church— the church that lets you become an "ordained minister" by simply paying $5, and lets you become a "doctor of divinity" by paying just $20. Rev. Jaenisch is known as the "Archbishop in Charge of Keyboarding". For his next feat, he plans to make the computer perform divorces. He also uses the computer to persuade kids to come to church. He claims, "What better way to get kids into church than by letting them play with a computer? It's more interesting than praying."

For a while, "Reverend Apple" was retired, because Jaenisch couldn't interest enough couples. He complained, "It's not easy to convince people to get married by a computer. They don't think it's romantic." NBC television news and many newspapers wanted to interview him; but he became embarrassed, because he couldn't find enough couples. And anyway, he's a reverend only part-time. His main job is as an employment agent: he's supposed to help companies find programmers. He thought Reverend Apple's reputation would help him find programmers, but it didn't.

But Reverend Apple is finally starting to catch on. During its first eight months, it performed six marriages. Jaenisch says, "The first couple had nothing to do with computers professionally: the groom drove a tow-truck and was an hour late for the ceremony because he wanted to work overtime. But the second couple was *very* involved with computers: they even asked for a printout of the ceremony." For the sixth ceremony, the groom (who earns his living by fixing computer power supplies) says, "It was nice with our friends all gathered around the console, and someone brought champagne. But part of our vow was to never buy a home computer: we have to get away from machines *some*time."

LOVE BUG (1980)

You can buy a *Love Bug*. It's a small computerized box that you put in your pocket. You feed the box information about your personality. When you walk through a singles bar, if you get near a person of the opposite sex who's compatible and has a Love Bug also, your Love Bug beeps. As you and the other person get closer and closer, the Love Bugs beep to each other ever more violently. The more violently your Love Bug beeps, the closer you are to your ideal partner.

Using a Love Bug to find a date is like using a Geiger counter to find uranium. The louder the Love Bug beeps, the louder your heart will pound.

SELECTROCUTION (1980)

If you don't like the Love Bug, how about a love *billboard*? One company sells love billboards to singles bars.

Each person who enters the bar gets to wear a gigantic name tag, which shows the person's initials. For example, since I'm Russ Walter, my tag would say, in gigantic letters, "RW". If I see an attractive gal whose tag says "JN", and I like her smile, I tell the person who operates the billboard. A few seconds later, a gigantic computerized billboard hanging over the entire crowd flashes this message:

FOR JN FEMALE: YOU HAVE A NICE SMILE--RW MALE

Everybody in the bar sees my message. When the gal of my dreams, "JN female", sees it, she hunts for "RW male", and we unite in computerized joy.

That's great for bashful people, like me, who'd rather pass notes than face a stranger unprepared.

It's called *Selectrocution*, because it gives your social life an electronic tingle that ends all your problems.

INTERLUDE (1980)

The most provocative sex program is *Interlude*. It interviews both you and your lover, then tells you what sexual activities to perform. Some of the activities are quite risqué. (Puritans think the program should be called "Inter Lewd".)

The program runs on your Radio Shack or Apple computer. (The explicit full-color ad shows a half-clad girl on satin sheets caressing her Apple.)

The program is based loosely on Masters-and-Johnson sexual therapy. It interviews each person separately— and privately— and then recommands a sexual interlude.

During the interview, the computer asks you questions such as:

HOW LONG WOULD YOU LIKE THE INTERLUDE TO LAST?

You can choose any length of time, from "several seconds" to "several days". If you choose "several seconds", the computer will recommend that while driving home from a party, you put your lover's finger in your mouth and seductively caress it with your tongue. If you choose "several days", the computer will recommend that you tell your lover to meet somebody at the airport. When your lover arrives at the airport, your lover finds *you* there instead; you tell your lover that the two of you are going on a surprise vacation, and that you've already bought the tickets.

The computer also asks questions such as:

DO YOU LIKE SURPRISES?

You have several choices: you like to *give* surprises; you like to *be* surprised; or you don't like surprises at all. If you like to *be* surprised, and your lover likes to *give* surprises, the computer will tell you to leave the room; after you've left, the computer will give secret information to your lover, about the best way to surprise you.

The computer asks which parts of the body you like. (One of the choices is: "buttocks".) The computer also asks which kinds of accessories you like. (One of the choices is: "whips and chains".) The computer asks whether you want the interlude to occur "immediately" or "later": if you say "later", the computer will recommend that you buy elaborate props, to make the interlude fancier.

Some of the interludes are unusual. For example, if you're a woman and want to surprise your husband, the computer recommends that you call his office and invite him home for lunch. When he arrives, he finds all the shades pulled down: you do a nude dance on the table, then sit down to eat.

During the interview, the computer's questions are often corny. For example, the computer asks:

IF YOUR INTERLUDE WERE ON T.V., WHAT SHOW WOULD IT RESEMBLE?

Sample choices are "Three's Company", "Roots", and "a commercial". If you say "Roots", the computer says "heavy!" If you say "a commercial", the computer says "yecch!"

The computer asks how much sex you'd like. If you say "lots!" but your lover says the opposite, the computer will recommend that you take a cold shower, to cool your hot passion.

If you've been married for at least twenty years, you'd probably like to change a few things about your sex life but are afraid to tell your spouse that you've been less than thrilled. You'd like an intermediary to whom you can express your anxieties and who will pass the message to your spouse gently. The Interlude program acts as that intermediary, in a playful way.

Interlude's programmer says he created it because he was tired of hearing people wonder what to do with their personal computers. Once you've tried the Interlude program, your personal computer will suddenly become *very* personal!

It's rated R. To avoid an X rating, it insists on having one man and one woman: it doesn't permit homosexuality or group sex or masturbation. Sorry!

Interlude costs $18.95 on tape, $21.95 on disk, plus $1.75 shipping. It works on Apple and on Radio Shack's model 1. Send check payable to Interlude (at 10635 Richmond Ave., Houston, TX 77042; if you live in Texas, phone 800-392-2348 ext. 306; out-of-state phone 800-231-5768 ext. 306).

The program came out in May, 1980. Within a year, ten thousand copies were sold.

PORNOPOLY (1980)

To have an orgy, try this trick. Invite your friends over for a "game". Tell them it's a computerized version of Monopoly. When they arrive, surprise them by telling them they'll play *Pornopoly*, the computerized version of Monopoly that's rated X.

In Pornopoly, you buy and sell property, just as in Monopoly. But the streets have names such as Bedroom Avenue, Horny Avenue, Hot Jugs Avenue, Jock Strap Place, and Orgasm Railroad. You get penalty cards such as: name 7 four-letter words that rhyme with duck. You might be told to play doctor, and conduct a physical examination of another player...or remove the pants of your favorite player by using only your teeth. When a player lands on a monopoly that you own, the player must take a drink, remove an article of clothing, kiss you, give you a free feel, or strip completely for two turns. At the end of the game, whoever remains dressed is the winner.

This successful program has been featured on national television. Copies have been requested by Hugh Hefner, Johnny Carson, Rona Barrett, an army chaplain, and a dozen foreign countries. To add your own name to that list, contact Computer Consultants of Iowa Ltd. (Box 427, Marion, Iowa 52302, 319-373-1306).

Pornopoly costs $30, but the company doesn't accept money: it accepts only Master Charge, Visa, and COD. If you're a kid, tough luck: the company says, "This is an adult party game rated XXX and some people may find it offensive."

Among the offended is a New Orleans grandmother who read an article about the program and wrote this note to the company: "Thanks to you, I intend to start contributing to Moral Majority, something I have avoided until now."

The program is available for Radio Shack, Apple, Commodore, and Atari computers. *Infoworld* (the microcomputing industry's scandal sheet) criticises the Atari version for its poor graphics, vague manual, and occasional bugs. If you try the Radio Shack, Apple, or Commodore version, tell me how you like it. And can I play?

ANALYZE YOURSELF

DEATH

Terrence Lukas wrote a program that predicts when you'll die. The program makes the computer ask for your age and sex; then it asks about the life and health of your parents and grandparents, your weight, your personal habits (smoking, drinking, exercise, and sleep), your history of medical check-ups, your social class (your education, occupation, and income), and your lifestyle: urban or rural, single or married, aggressive or passive, and whether you use seat belts. The computer combines all that information, to tell you when you'll probably die.

The program uses the latest statistics from life-insurance companies and from medical research. He wrote the program at the University of Illinois Medical Center.

Running the program is fun. Each time you answer a question, the computer tells you how your answer affects its prediction. You see its prediction bob up and down, until the questions finally end, and the computer gives you its final prediction of when you'll die. It's like watching the early returns for a Presidential Election, except the topic is you!

The computer pops out with surprising comments, based on the latest medical research. Here are some comments the computer prints:

Professionals usually live longer, except musicians, architects, and pharmacists. Why this is true is unknown.

Cooks, chefs, bakers, and other people who work at jobs associated with overeating have a lower life expectancy.

Adults that sleep too much use too many hours in nonphysical activity. They may be unhappy and sleep as an escape, or may be ill. Depressed people have shorter life expectancies.

Moderate drinking (up to two drinks per day) reduces stress and aids digestion. Heavy drinking, however, produces physiological damage. As for teetotalers, they may have rather rigid value systems and may undergo stress in maintaining them.

The program is on pages 34-36 of the November 1977 issue of *Kilobaud Microcomputing Magazine*.

BRAINWAVES

A computer has been programmed to read your mind, by analyzing your brainwaves. A newspaper article described the program dramatically: you're an airplane pilot...your plane is going to crash...but you think "Up!", and the plane automatically goes back up!...because the plane is run by a computer that's reading your brainwaves!

But what if the pilot is sadistic, and thinks "Down"?

Anyway, the program isn't perfected yet. When the computer tries to distinguish brainwaves that mean *up* from brainwaves that mean *down*, it gets the right answer 75% of the time. In other words, 25% of the time it goofs. I'd hate to be in a plane controlled by a computer that was having a bad day!

Recently, the U.S. Government has shifted the emphasis of this research. The new emphasis is more practical: to make the computer ring a buzzer, when the pilot's brainwaves indicate the pilot is daydreaming.

HOROSCOPES

If you hate horoscopes as much as I do, you'll love Adrian Thornton's program. It makes the computer print fake horoscopes. The computer asks when you were born, computes your sign, and then prints an enchanting remark.

For example, if your sign turns out to be Capricorn, the computer will print:

You are conservative and afraid of taking risks. You don't do much of anything and are lazy. There has never been a Capricorn of any importance. Capricorns should avoid standing still too long, as they tend to take root and become trees.

If you're a Virgo, the computer will print:

You are the logical type and hate disorder. This nitpicking is sickening to your friends. You are cold and unemotional and sometimes fall asleep while making love. Virgos make good bus drivers.

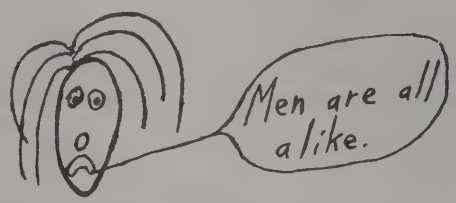
You'll enjoy seeing your own "horror"scope. The complete program is on pages 48-49 of the February 1979 issue of *Kilobaud Microcomputing Magazine*.

PSYCHOTHERAPY

In 1966, Joseph Weizenbaum programmed the computer to act as a psychotherapist. On the keyboard, you type whatever thoughts come to your mind, and the computer will give you therapy. When typing on the terminal, feel free to use whatever words you wish— you don't have to use a "computer language".

The most amazing thing about the program is that it does a decent job of therapy, even though the computer doesn't know what you're talking about! Weizenbaum's program contains the definitions of only a handful of words; but if you use words the computer doesn't know, the computer does an amazingly good job of "faking". For example, if you say, "My junkapoo is like my iliabop", the computer will continue the conversation by asking you, "In what way?" even though the computer doesn't know what a junkapoo or an iliabop is.

Here's a conversation between his computer and a patient.



PATIENT: *Men are all alike.*
COMPUTER: *In what way?*

PATIENT: *They're always bugging us about something or other.*
COMPUTER: *Can you think of a specific example?*

PATIENT: *Well, my boyfriend made me come here.*
COMPUTER: *Your boyfriend made you come here.*

PATIENT: *He says I'm depressed much of the time.*
COMPUTER: *I am sorry to hear you are depressed.*

PATIENT: *It's true. I am unhappy.*
COMPUTER: *Do you think coming here will help you not to be unhappy?*

PATIENT: *I need some help, that much seems certain.*
COMPUTER: *What would it mean to you if you got some help?*

PATIENT: *Perhaps I could learn to get along with my mother.*
COMPUTER: *Tell me more about your family.*

PATIENT: *My mother takes care of me.*
COMPUTER: *Who else in your family takes care of you?*

PATIENT: *My father.*
COMPUTER: *Your father.*

PATIENT: *You are like my father in some ways.*
COMPUTER: *What resemblance do you see?*

PATIENT: *You are not very aggressive, but I think you don't want me to notice that.*
COMPUTER: *What makes you think I am not very aggressive?*

PATIENT: *You don't argue with me.*
COMPUTER: *Why do you think I don't argue with you?*

PATIENT: *You are afraid of me.*
COMPUTER: *Does it please you to believe I am afraid of you?*

PATIENT: *My father is afraid of everybody.*
COMPUTER: *What else comes to your mind when you think of your father?*

PATIENT: *Bullies.*
COMPUTER: *Does that have anything to do with the fact that your boyfriend made you come here?*

Like the famous psychotherapist Carl Rogers, the computer asks probing questions that make the patient open up, but never gives advice: the patient must discover the truth for herself.

I'll explain exactly how the program works, so you can become a psychotherapist yourself, and get hundreds of thousands of dollars by bilking your patients.

The computer begins by replacing some of the patient's words:

Word that the patient typed	Replacement
mom	mother
dad	father
dont	don't
cant	can't
wont	won't
dreamed	dreamt
dreams	dream
I	you@
me	you
you	I
my	your
your	my
myself	yourself
yourself	myself
I'm	you're
you're	I'm
am	are@
were	was

For example, the sentence *Well, my boyfriend made me come here* becomes *Well, your boyfriend made you come here.*

The computer hunts through the revised sentence or paragraph for one of these *keywords....*

CATEGORY 8: computer, computers, machine, machines
CATEGORY 7: name
CATEGORY 6: alike, like, same
CATEGORY 5: remember
CATEGORY 4: dreamt
CATEGORY 3: dream, if
CATEGORY 2: everybody, everyone, nobody, noone, was, your
CATEGORY 1: always
CATEGORY 0: are, are@, because, can, certainly, deutsch, espanol, francais, hello, how, I, I'm, italiano, maybe, my, no, perhaps, sorry, what, when, why, yes, you@, you're

If the computer finds *several* of those keywords, it chooses the one in the highest category; if they lie in the same category, it chooses the one the patient typed first.

Then it looks up the keyword in this phrasebook:

Patient's keyword	Computer's reply
alike	In what way?
always	Can you think of a specific example?
are	
are I...	Why are you interested in whether I am...or not?
are...	Did you think they might not be...?
are@	
are you...	Do you believe you are...?
are	Why do you say "am"?
because	Is that the real reason?
can	
can I...	You believe I can..., don't you?
can you...	Whether or not you can...depends on you more than on me.
certainly	You seem quite positive.
computer	Do computers worry you?
computers	Do computers worry you?
deutsch	I am sorry, I speak only English.
dream	What does that dream suggest to you?
dreamt...	Really,...?
espanol	I am sorry, I speak only English.
everybody	Really, everybody?
everyone	Really, everone?
hello	How do you do— please state your problem.
how	Why do you ask?
I	
I remind you of	In what way?
I are...	What makes you think I am...?
I...you	Why do you think I...you?
I	We were discussing you— not me.
if...	Do you think it's likely that...?
I'm...	What makes you think I am...?
italiano	I am sorry, I speak only English.
like	
(am,is,are,was)...like	In what way?
like	(get a different keyword)
machine	Do computers worry you?
machines	Do computers worry you?
maybe	You don't seem quite certain.
my...	Why are you concerned over my...?
name	I am not interested in names.
no	Are you saying "no" just to be negative?
nobody	Really, nobody?
noone	Really, noone?
perhaps	You don't seem quite certain.
remember	
you remember...	Do you often think of...?
do I remember...	Did you think I would forget...?
remember	(get a different keyword)
same	In what way?
sorry	Please don't apologize.
was	
was you...	What if you were...?
you was	Were you really?
was I...	Would you like to believe I was...?
was	(get a different keyword)
what	Why do you ask?
when	Why do you ask?
why	
why don't I...	Do you believe I don't...?
why can't you...	Do you think you should be able to...?

yes	You seem quite positive.
you@	
you (want,need)...	What would it mean to you if you got...?
you are...(sad,unhappy,depressed,sick)	I am sorry to hear you are (sad,etc.).
you are...(happy,elated,glad,better)	How have I helped you to be (happy,etc.)?
you (feel,think,believe,wish) you	Do you really think so?
you (feel,think,believe,wish)...I	(use the keyword "I" instead)
you are...	Is it because you are...that you came to me?
you (can't,cannot)...	How do you know you can't...?
you don't...	Don't you really...?
you feel	Tell me more about such feelings.
you...I	Perhaps in your fantasy we...each other.
your	
your...(mother,father,sister,brother,wife,children)	Tell me more about your family.
your...	Your....
you're	(treat as "you@ are")

For example, if the keyword is *sorry*, the computer looks up *sorry* in the phrasebook, which says to print "Please don't apologize."

Suppose the patient types, "If the job is lousy, he'll die." The keyword is *if*. In the phrasebook, *if* is followed by three dots, which stand for the part of the clause that comes after *if*, which is "the job is lousy". (The computer figures out where the clause ends by looking at the punctuation.) The phrasebook says to print "Do you think it's likely that the job is lousy?"

The symbol @ serves just to locate the correct keyword in the phrasebook. Thereafter, it's ignored.

Here's what happens if the keyword is *you@*. After locating *you@* in the phrasebook, the computer ignores the @. If the patient's revised sentence contains *you want...* or *you need...*, the computer prints "What would it mean to you if you got...?" If the patient's sentence contains *you are...sad*, the computer prints "I am sorry to hear you are sad."

For each reply in the phrasebook, Weizenbaum stored a list of alternatives. For example, here are the alternatives to "Please don't apologize":

Apologies are not necessary.
What feelings do you have when you apologize?
I've told you that apologies are not required.

While chatting with the patient, the computer keeps track of which replies it has printed already, and uses the alternatives to avoid repetition.

If the patient's statement doesn't contain a keyword, the computer may give one of these replies:

I am not sure I understand you fully.
Please go on.
What does that suggest to you?
Do you feel strongly about discussing such things?

Or it may take a second look at earlier parts of the conversation, retrieve a clause that contained *your...*, and print one of these replies:

Let's discuss further why your....
Earlier you said your....
Does that have anything to do with the fact that your...?

For example, it may retrieve the clause *Your boyfriend made you come here* and print:

Does that have anything to do with the fact that your boyfriend made you come here?

The program was written at the Massachusetts Institute of Technology (MIT). The programmer, Joseph Weizenbaum, named the program "Eliza", to honor Eliza Doolittle—the woman in *My Fair Lady* and *Pygmalion*, who learned to speak English and have a good conversation.

Some people think Weizenbaum's program shows that computers can communicate as well as psychotherapists. But Weizenbaum himself holds the opposite view: he believes the program shows that psychotherapists communicate as poorly as computers.

After Weizenbaum, other programmers invented improvements. For example, in one of the improved versions, whenever the computer doesn't understand you, it reacts like the typical rich psychotherapist— it bluffs and says:

I understand fully. Please go on.

The computer never really understands what you say. For example, in one version, if you say "My, my, my!", the computer will reply:

Tell me more about your your your.

Radio Shack's headquarters phoned me, and asked for permission to use my explanation of Weizenbaum's program. I agreed, whereupon Radio Shack invented and sold a version that makes the computer *talk*— by using a voice synthesizer. You'll freak out when you hear the computer's voice giving you therapy. Radio Shack's program costs $15, comes on tape, and requires a model 1 or model 3 TRS-80. To hear the voice, you must buy the voice synthesizer. (But even if you don't buy the voice synthesizer, the program will impress you.)

The cheapest way to get a version of Eliza is to buy a book called *BASIC Computer Games Volume 2 (TRS-80 edition)*. Get it at your local Radio Shack store. It costs just $7 and includes 84 programs— one of which is a version of Eliza written in BASIC. That version of Eliza isn't as good as the original; but since it's cheap and written in BASIC, you'll enjoy buying it for self-study. Here's part of a conversation between that version and a patient:

PATIENT: I am not the greatest conversationalist.
COMPUTER: Do you believe it is normal to be not the greatest conversationalist?

PATIENT: Yes.
COMPUTER: I see.

PATIENT: How can you see without any eyes?
COMPUTER: Don't you believe that I can see without any eyes?

PATIENT: No. So prove it.
COMPUTER: Are you saying "no" just to be negative?

PATIENT: Absolutely not.
COMPUTER: You are being a bit negative.

PATIENT: So you want to fight, do you?
COMPUTER: We were discussing you— not me.

PATIENT: Don't back down, friend.
COMPUTER: Why do you bring up the topic of friends?

PATIENT: I don't have any friends.
COMPUTER: Don't you really have any friends?

PATIENT: No, not really.
COMPUTER: Why not?

Unfortunately, most versions of Eliza are passive. The typical Eliza program works well only if the patient's talkative. If the patient doesn't say much, the program can't react to much and is useless.

Recently, John Holland has invented a better version of Eliza. His version is more aggressive. He calls it "Analiza 2" (because it "analizas" the patient). If the patient's too quiet or too boring or too uninformative, Analiza 2 makes the computer give the patient a "jolt", by asking an electrifying question or making a ridiculous Zen-like statement. For example, the computer might ask, "Did you ever try to commit suicide with a hammer?" Here's a list of the computer's favorite "jolts":

Nasty questions the computer asks
Did you ever try to commit suicide with a hammer?
Is your interest in sex still not very strong?
Do you want to know what I suspect?

Prying questions the computer asks
Tell me, what is your worst fear?
Why don't you tell me about your first romance?
I want to know about your lifestyle.
Tell me what is troubling you.
Honestly now, what is your biggest concern?

Ridiculous statements the computer makes
With strong drugs you can learn to control your concern!
You certainly are curious!
And you are obstinate!

Complaints the computer makes
You seem inhibited.
This isn't really very exciting.
Getting started is often difficult.

Comments the computer makes about timing and complexity
Well, time is almost up. Is there anything else you want to discuss?
I hate to bring this up, but you still owe me $53.00 from your last visit!
Last time you were extremely inquisitive. Have you abandoned seeking simple solutions?
Last time you were difficult to understand. Were you trying to confuse me?
Let's keep it simple this time.
Say, I am having trouble with your long sentences. Keep it simple. I am only human.

Yes, your patient will love to hear the computer says it's "only human". You can get Analiza 2 for $50 from Supersoft (Box 1628, Champaign, IL 61820, 217-359-2112). It runs on most computers that have the CP/M operating system.

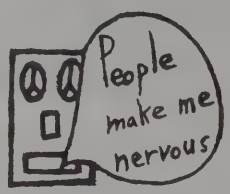
What if the world becomes full of computerized psychoanalysts, and there aren't enough patients to keep them all busy? Kenneth Colby solved that problem: in 1971 at Stanford University, he programmed a computer to be a crazy patient. Specifically, he programmed his computer to be a paranoid schizophrenic. In the following conversations, psychiatrists interrogate his computer.

CONVERSATION 1

PSYCHIATRIST: *How long have you been in the hospital?*
COMPUTER: *About a week.*

PSYCHIATRIST: *Are you depressed?*
COMPUTER: *No.*

PSYCHIATRIST: *Were you taking drugs?*
COMPUTER: *No.*



PSYCHIATRIST: *What is your problem?*
COMPUTER: *People make me nervous.*

PSYCHIATRIST: *How do they do that?*
COMPUTER: *They stare at me.*

PSYCHIATRIST: *Why?*
COMPUTER: *Maybe they don't like the way I look or something.*

PSYCHIATRIST: *Do you look unusual in some way?*
COMPUTER: *Why are you interested in my looks?*

PSYCHIATRIST: *Because you said people stare at you.*
COMPUTER: *Well, sometimes.*

PSYCHIATRIST: *How do you get along with people?*
COMPUTER: *I don't communicate very much with the others.*

PSYCHIATRIST: *Why not?*
COMPUTER: *They don't realize what's going on most of the time.*

PSYCHIATRIST: *What is going on?*
COMPUTER: *Do you know how the underworld operates?*

CONVERSATION 2

PSYCHIATRIST: *How are you today?*
COMPUTER: *OK.*

PSYCHIATRIST: *What is your name?*
COMPUTER: *My name is Frank Smith.*

PSYCHIATRIST: *How old are you?*
COMPUTER: *Twenty-eight.*

PSYCHIATRIST: *Are you married?*
COMPUTER: *No.*

PSYCHIATRIST: *What sort of work do you do?*
COMPUTER: *I work in the post office.*

PSYCHIATRIST: *Why are you in the hospital?*
COMPUTER: *I shouldn't be here.*

PSYCHIATRIST: *Who brought you here?*
COMPUTER: *The police.*

PSYCHIATRIST: *What trouble did you have with the police?*
COMPUTER: *Cops don't do their job.*

PSYCHIATRIST: *What do they do instead?*
COMPUTER: *Cops arrest the wrong people.*

PSYCHIATRIST: *They arrest the right ones sometimes.*
COMPUTER: *The law should get those Italian crooks.*

PSYCHIATRIST: *Are you Italian?*
COMPUTER: *My background is British but I was born in this country.*

PSYCHIATRIST: *Do you know any Italian crooks?*
COMPUTER: *I try to avoid the underworld.*

PSYCHIATRIST: *Are you afraid of the underworld?*
COMPUTER: *The Mafia are the worst gangsters.*

PSYCHIATRIST: *Are you connected with the Mafia?*
COMPUTER: *You know, they know me.*

PSYCHIATRIST: *How do they know you?*
COMPUTER: *They spy on me.*

PSYCHIATRIST: *Why do they?*
COMPUTER: *They want to get false evidence to frame me.*

PSYCHIATRIST: *What do they have against you?*
COMPUTER: *They are out to get me.*

PSYCHIATRIST: *What for?*
COMPUTER: *I have incriminating evidence against the Mafia.*

PSYCHIATRIST: *Are you sure of all this?*
COMPUTER: *You don't believe me?*

CONVERSATION 3

PSYCHIATRIST: *Tell me something about yourself.*
COMPUTER: *I went to the track at Bay Meadows a while back.*

PSYCHIATRIST: *What happened there?*
COMPUTER: *There's a lot of crooked gambling there.*

PSYCHIATRIST: *Why do you think the gambling is crooked?*
COMPUTER: *I have worried that gangsters have too much power in this country.*

CONVERSATION 4

PSYCHIATRIST: *How do you like the hospital?*
COMPUTER: *I don't communicate very much with others.*

PSYCHIATRIST: *Do other people at the hospital get on your nerves?*
COMPUTER: *I shouldn't have come here.*

PSYCHIATRIST: *Why do you say you shouldn't have come here?*
COMPUTER: *They made me come here.*

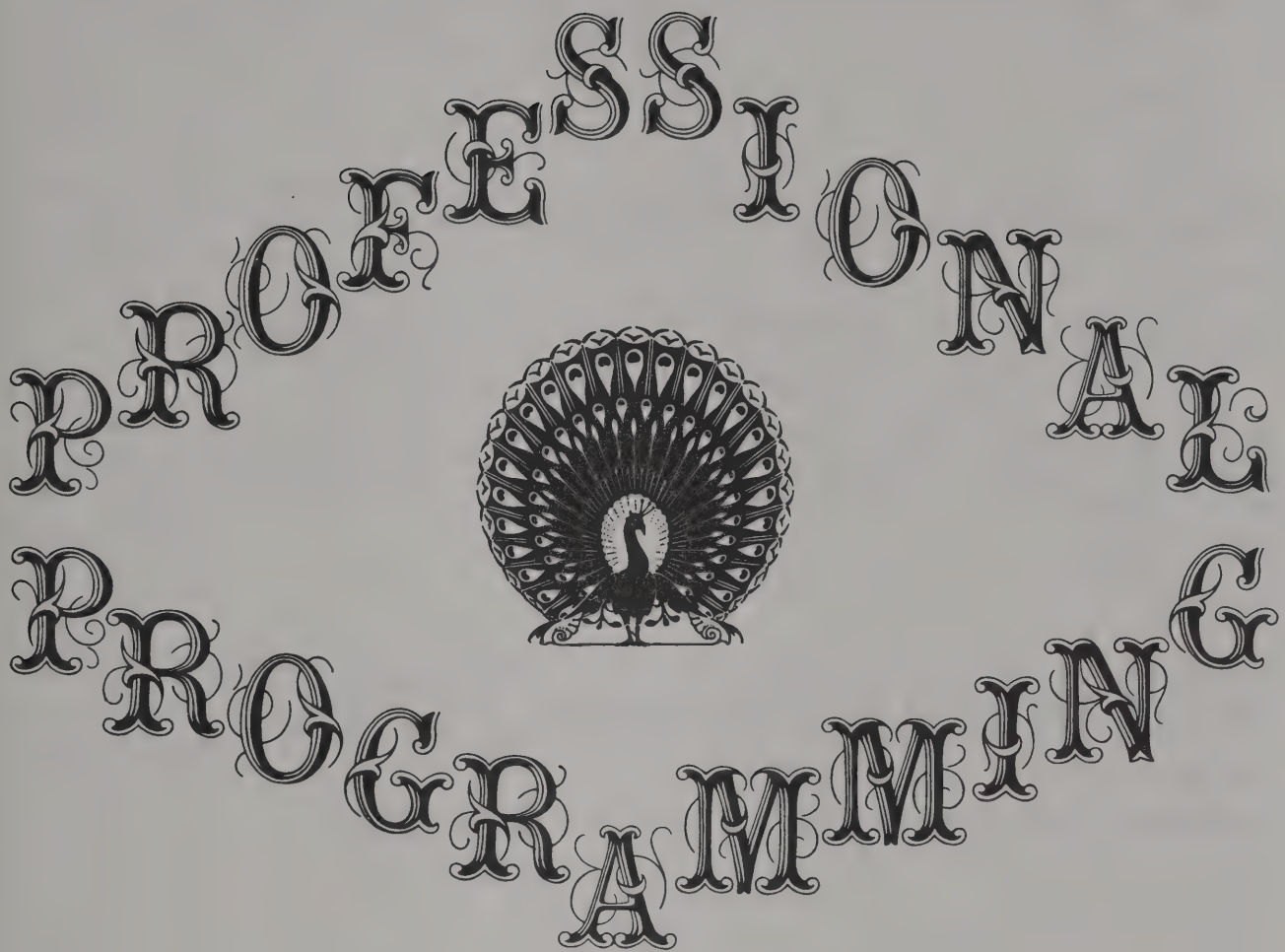
The computer's response depends on how much FEAR, ANGER, and MISTRUST it has at the moment. Those three variables are affected by what the psychiatrist says. For example, if the psychiatrist gives the computer a compliment, the FEAR, ANGER, and MISTRUST usually decrease. But if MISTRUST is already high, the computer interprets the compliment as a sly attempt at pacification, and its ANGER increases instead of decreasing. ANGER can also be increased by questions that humiliate (*Tell me about your sexlife*) or imply subjugation (*Perhaps you should stay in the hospital longer*).

To prove his computer is paranoid, Colby had 8 psychiatrists interview it and also interview paranoid humans via teletypewriters. Transcripts of all the interviews were sent to psychiatrists around the country, who were asked to judge whether each interview was with a human or with the computer. The psychiatrists were unable to tell the difference: only 51% of their guesses were correct.

Some computerniks got the "brainstorm" or hooking Weizenbaum's computer to Colby's, to see whether the computerized psychotherapist could cure the computerized schizophrenic. The experiment was a disaster: both computers were so passive that the discussion rapidly degenerated into trivia.

But so do conversations between humans!

ご休憩ご自由に 6


PROFESSIONAL
PROGRAMMING

SUBSCRIPTS

SIMPLE SUBSCRIPTS

Instead of being a single number, X can be a whole *list* of numbers, like this:

$$X=\begin{pmatrix} 57.2 \\ -8.3 \\ 476 \\ 2.008 \\ -19 \\ 372.402 \\ 0 \\ 215.6 \end{pmatrix}$$

Here's how to make X be that list of numbers....

Begin your program by saying:

```
10 DIM X(8)
```

That says X will be a list of 8 numbers. DIM means *dimension*; the line says the dimension of X is 8.

Next, tell the computer what numbers are in X. Type these lines:

```
20 X(1)=57.2
30 X(2)=-8.3
40 X(3)=476
50 X(4)=2.008
60 X(5)=-19
70 X(6)=372.402
80 X(7)=0
90 X(8)=215.6
```

Line 20 says: X's first number is 57.2. Line 30 says X's second value is -8.3. The remaining lines define the other numbers in X.

If you'd like the computer to print all those numbers, type this:

```
100 FOR I = 1 TO 8: PRINT X(I): NEXT
```

That means: print all the numbers in X. The computer will print:

```
 57.2
-8.3
 476
 2.008
-19
 372.402
 0
 215.6
```

In that program, line 20 talks about X(1). Instead of saying X(1), math books say:

X_1

The "1" is called a *subscript*. Similarly, in line 30, which says "X(2)=-8.3", the number 2 is a subscript. Some programmers pronounce line 30 like this: "X subscripted by 2 is -8.3". To be briefer, most programmers say this instead: "X sub 2 is -8.3". Some programmers simply say: "X 2 is -8.3".

In that program, X is called an *array* (or *matrix*). Definition: an *array* (or *matrix*) is a variable that has subscripts.

That program said X(1) is 57.2, and X(2) is -8.3, and so on. This program does the same thing, more briefly:

```
10 DIM X(8)
20 DATA 57.2, -8.3, 476, 2.008, -19, 372.402, 0, 215.6
30 FOR I = 1 TO 8: READ X(I): NEXT
40 FOR I = 1 TO 8: PRINT X(I): NEXT
```

Line 10 says X will be a list of 8 numbers. Line 20 contains a list of 8 numbers. Line 30 makes the computer READ those numbers, and call them X. Line 40 makes the computer print them.

In that program, the first three lines say:

DIM
DATA
FOR I

Most practical programs begin with those three lines.

Let's make the program fancier, by adding extra lines. This line makes the computer add the 8 numbers together and print the sum:

```
50 PRINT X(1)+X(2)+X(3)+X(4)+X(5)+X(6)+X(7)+X(8)
```

That line makes the computer print:

```
 1095.91
```

Let's make the computer print the numbers in reverse order (starting with the 8th number, and ending with the first). In other words, let's make the computer print X(8), then print X(7), then print X(6), etc. To do that, you could say:

```
60 PRINT X(8)
70 PRINT X(7)
80 PRINT X(6)
```

etc.

But this way is shorter:

```
60 FOR I = 8 TO 1 STEP -1: PRINT X(I): NEXT
```

Line 60 prints:

```
 215.6
 0
 372.402
-19
 2.008
 476
-8.3
 57.2
```

STRANGE EXAMPLE

Getting tired of X? Then pick another letter! For example, you can play with Y:

Silly, useless program	What the program means
`10 DIM Y(5)`	Y will be a list of 5 numbers.
`20 FOR I = 2 TO 5`	
`30  Y(I)=I*100`	Y(2) is 200. Y(3) is 300. Y(4) is 400. Y(5) is 500.
`40 NEXT`	
`50 Y(1)=Y(2)-3`	Y(1) is 200-3; so Y(1) is 197.
`60 Y(3)=Y(1)-2`	Y(3) changes. It becomes 197-2, which is 195.
`70 FOR Y = 1 TO 5: PRI NT Y(I): NEXT`	Print Y(1), Y(2), Y(3), Y(4), and Y(5).

Line 70 prints:

```
 197
 200
 195
 400
 500
```

PROBLEMS AND SOLUTIONS

Suppose you want to analyze 50 numbers. Begin your program by saying:

```
10 DIM X(50)
```

Then type the 50 numbers, as data, like this:

```
20 DATA etc.
30 DATA etc.
40 DATA etc.
```

Tell the computer to READ the data:

```
100 FOR I = 1 TO 50: READ X(I): NEXT
```

After line 100, you have many choices, depending on which problem you want to solve....

Problem: print all the values of X. Solution:

```
110 FOR I = 1 TO 50
120    PRINT X(I)
130 NEXT
```

Problem: print all the values of X, in reverse order. Solution:

```
110 FOR I = 50 TO 1 STEP -1
120    PRINT X(I)
130 NEXT
```

Problem: print the sum of all the values of X. In other words, print X(1)+X(2)+X(3)+...+X(50). Solution: start the sum at 0—

```
110 S=0
```

and then increase the sum, by adding each X(I) to it:

```
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT
```

Finally, print the sum:

```
150 PRINT "THE SUM OF ALL THE NUMBERS IS";S
```

Problem: find the average value of X. In other words, find the average of the 50 numbers. Solution: begin by finding the sum—

```
110 S=0
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT
```

and then divide the sum by 50:

```
150 PRINT "THE AVERAGE IS";S/50
```

Problem: find out whether any of the values of X is 79.4. In other words, find out whether 79.4 is a number in the list. Solution: if X(I) is 79.4, print "YES"—

```
110 FOR I = 1 TO 50
120    IF X(I)=79.4 THEN PRINT "YES, 79.4 IS IN THE LIST": END
130 NEXT
```

otherwise, print "NO":

```
140 PRINT "NO, 79.4 IS NOT IN THE LIST"
```

Problem: in the list of numbers, count how often the number 79.4 appears. Solution: start the counter at zero—

```
110 C=0
```

and increase the counter each time you see the number 79.4:

```
120 FOR I = 1 TO 50
130    IF X(I)=79.4 THEN C=C+1
140 NEXT
```

Finally, print the counter:

```
150 PRINT "THE NUMBER 79.4 APPEARS";C;"TIMES"
```

Problem: print all the values of X that are negative. In other words, print all the numbers that have minus signs. Solution: begin by announcing your purpose—

```
110 PRINT "HERE ARE THE VALUES THAT ARE NEGATIVE:"
```

and then print the values that are negative; in other words, print each X(I) that's less than 0:

```
120 FOR I = 1 TO 50
130    IF X(I)<0 THEN PRINT X(I)
140 NEXT
```

Problem: print all the values of X that are "above average". Solution: find the average, and make A the average—

```
110 S=0
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT
150 A=S/50
```

then announce your purpose:

```
160 PRINT "THE FOLLOWING VALUES ARE ABOVE AVERAGE:"
```

Finally, print the values that are above average; in other words, print each X(I) that's greater than A:

```
170 FOR I = 1 TO 50
180    IF X(I)>A THEN PRINT X(I)
190 NEXT
```

Problem: find the biggest value of X. In other words, find which of the 50 numbers is the biggest. Solution: let B stand for the biggest number. Begin by tentatively setting B equal to the first number—

```
110 B=X(1)
```

but if another number is bigger than that B, change B:

```
120 FOR I = 2 TO 50
130    IF X(I)>B THEN B=X(I)
140 NEXT
```

Afterwards, print B:

```
150 PRINT "THE BIGGEST NUMBER IN THE LIST IS";B
```

Problem: find the smallest value of X. In other words, find which of the 50 numbers is the smallest. Solution: let S stand for the smallest number. Begin by tentatively setting S equal to the first number—

```
110 S=X(1)
```

but if another number is smaller than S, change S:

```
120 FOR I = 2 TO 50
130    IF X(I)<S THEN S=X(I)
140 NEXT
```

Afterwards, print S:

```
150 PRINT "THE SMALLEST NUMBER IN THE LIST IS";S
```

Problem: find out whether the values of X are in strictly increasing order. In other words, find out whether the following statement is true: X(1) is a smaller number than X(2), which is a smaller number than X(3), which is a smaller number than X(4), etc. Solution: if X(I) is *not* smaller than X(I+1), print "NO"—

```
110 FOR I = 1 TO 49
120    IF X(I)>=X(I+1) THEN PRINT "NO, THE LIST IS NOT IN STRICTLY INCREASING OR
DER": END
130 NEXT
```

otherwise, print "YES":

```
140 PRINT "YES, THE LIST IS IN STRICTLY INCREASING ORDER"
```

Test yourself: look at those problems again, and see whether you can figure out the solutions *without peeking at the answers*.

MULTIPLE ARRAYS

Suppose your program involves three lists of numbers. Suppose the first list is called A; the second list is called B; and the third list is called X. Suppose A's list contains 18 numbers, B's list contains 57 numbers, and X's list contains just 3 numbers. To say all that, begin your program with this statement:

```
10 DIM A(18), B(57), X(3)
```

DOUBLE SUBSCRIPTS

You can make X be a *table* of numbers, like this:

$$X=\begin{pmatrix} 57 & 8.4 & -6 \\ 1000 & 0 & 7.77 \end{pmatrix}$$

Here's how to make X be that table....

Begin by saying:

```
10 DIM X(2,3)
```

That says X will be a table, having 2 rows and 3 columns.

Then tell the computer what numbers are in X. Type these lines:

```
20 X(1,1)=57
30 X(1,2)=8.4
40 X(1,3)=-6
50 X(2,1)=1000
60 X(2,2)=0
70 X(2,3)=7.77
```

Line 20 says: the number in X's first row and first column is 57. Line 30 says the number in X's first row and second column is 8.4. The remaining lines define the other numbers in X.

If you'd like the computer to print all those numbers, type this:

```
80 FOR I = 1 TO 2: FOR J = 1 TO 3: PRINT X(I,J),: NEXT: PRINT: NEXT
```

That means: print all the numbers in X. The computer should print:

```
 57              8.4             -6
 1000            0                7.77
```

In that program, X is called a *table* or a *doubly subscripted array*.

MULTIPLICATION TABLE

This program prints a multiplication table:

```
10 DIM X(10,4)
20 FOR I = 1 TO 10
30     FOR J = 1 TO 4
40             X(I,J)=I*J
50     NEXT
60 NEXT
70 FOR I = 1 TO 10: FOR J = 1 TO 4: PRINT X(I,J),: NEXT: PRINT: NEXT
```

Line 10 says X will be a table having 10 rows and 4 columns. Line 40 says the number in row I and column J is I*J; for example, the number in row 3 and column 4 is 12. Lines 20 and 30 make sure line 40 is done for *every* I and J, so *every* entry in the table is defined by multiplication. Line 70 prints the whole table:

```
 1              2              3              4
 2              4              6              8
 3              6              9              12
 4              8              12             16
 5              10             15             20
 6              12             18             24
 7              14             21             28
 8              16             24             32
 9              18             27             36
 10             20             30             40
```

Instead of multiplication, you can have addition, subtraction, or division: just change line 40.

Most programmers follow this tradition: the row's number is called I, and the column's number is called J. Line 40 obeys that tradition. Notice I comes before J in the alphabet; I comes before J in X(I,J); and "FOR I" comes before "FOR J" in lines 20-30. If you follow the I-before-J tradition, you'll make fewer errors.

SUMMING A TABLE

Suppose you want to analyze this table:

```
  32.7    19.4    31.6    85.1
  -8     402     -61       0
5106       -.2     0      -1.1
  36.9      .04    1      11
 777     666      55.44    2
   1.99    2.99    3.99    4.99
  50      40      30      20
  12      21      12      21
   0    1000       2     500
```

The table has 9 rows and 4 columns; so begin your program by saying:

```
10 DIM X(9,4)
```

Each row of the table becomes a row of the DATA:

```
11 DATA 32.7, 19.4, 31.6, 85.1
12 DATA -8, 402, -61, 0
13 DATA 5106, -.2, 0, -1.1
14 DATA 36.9, .04, 1, 11
15 DATA 777, 666, 55.44, 2
16 DATA 1.99, 2.99, 3.99, 4.99
17 DATA 50, 40, 30, 20
18 DATA 12, 21, 12, 21
19 DATA 0, 1000, 2, 500
```

Make the computer READ the data:

```
20 FOR I = 1 TO 9: FOR J = 1 TO 4: READ X(I,J): NEXT: NEXT
```

To make the computer print the table, say this:

```
30 FOR I = 1 TO 9: FOR J = 1 TO 4: PRINT X(I,J),: NEXT: PRINT: NEXT
```

Problem: find the sum of all the numbers in the table. Solution: start the at 0—

```
100 S=0
```

and then increase the sum, by adding each X(I,J) to it:

```
110 FOR I = 1 TO 9
120    FOR J = 1 TO 4
130           S=S+X(I,J)
140    NEXT
150 NEXT
```

Finally, PRINT the sum:

```
160 PRINT "THE SUM OF ALL THE NUMBERS IS";S
```

The computer will print:

```
THE SUM OF ALL THE NUMBERS IS 8877.84
```

Problem: find the sum of each row. In other words, make the computer print the sum of the numbers in the first row; and then print the sum of the numbers in the second row; and then print the sum of the numbers in the third row; etc. Solution: the general idea is—

```
100 FOR I = 1 TO 9
110    print the sum of row I
120 NEXT
```

Here are the details:

```
100 FOR I = 1 TO 9
110    S=0
111    FOR J = 1 TO 4
112           S=S+X(I,J)
113    NEXT
114    PRINT "THE SUM OF ROW";I;"IS";S
115 NEXT
```

The computer will print:

```
THE SUM OF ROW 1 IS 168.8
THE SUM OF ROW 2 IS 333
THE SUM OF ROW 3 IS 5104.7
etc.
```

Problem: find the sum of each column. In other words, make the computer print the sum of the numbers in the first column; and then print the sum of the numbers in the second column; and then print the sum of the numbers in the third column; etc. Solution: the general idea is—

```
100 FOR J = 1 TO 4
110    print the sum of column J
120 NEXT
```

Here are the details:

```
100 FOR J = 1 TO 4
110    S=0
111    FOR I = 1 TO 9
112        S=S+X(I,J)
113    NEXT I
114    PRINT "THE SUM OF COLUMN";J;"IS";S
115 NEXT
```

The computer will print:

```
THE SUM OF COLUMN 1 IS 6008.59
THE SUM OF COLUMN 2 IS 2151.23
THE SUM OF COLUMN 3 IS 75.03
THE SUM OF COLUMN 4 IS 642.99
```

In all the other examples we looked at, "FOR I" comes before "FOR J"; but in this unusual example, "FOR I" comes *after* "FOR J".

STRING ARRAYS

You've seen that X can be a list of numbers, or a table of numbers. Similarly, X$ can be a list of strings, or a table of strings. For example, you can make X$ be this table—

$$X\$=\begin{pmatrix} \text{"DOG"} & \text{"CAT"} & \text{"MOUSE"} \\ \text{"WOOF"} & \text{"MEOW"} & \text{"SQUEAK"} \\ \text{"HOTDOG"} & \text{"CATSUP"} & \text{"MOUSETARD"} \end{pmatrix}$$

by saying this—

```
10 DIM X$(3,3)
20 X$(1,1)="DOG"
30 X$(1,2)="CAT"
40 X$(1,3)="MOUSE"
50 X$(2,1)="WOOF"
etc.
```

or by saying this:

```
10 DIM X$(3,3)
20 DATA DOG,CAT,MOUSE,WOOF,MEOW,SQUEAK,HOTDOG,CATSUP,MOUSETARD
30 FOR I = 1 TO 3: FOR J = 1 TO 3: READ X$(I,J): NEXT: NEXT
```

SUBROUTINES

SIMPLE SUBROUTINES

Here's a simple program:

```
10 PRINT "THE"
20 GO TO 1000
30 PRINT "DEAD"
40 END

1000 PRINT "BIRD"
1010 PRINT "IS"
1020 GO TO 30
```

It makes the computer print:

```
THE
BIRD
IS
DEAD
```

The program consists of two parts. The main part (lines 10-40) is called the *main routine*; the bottom part (lines 1000-1020) is called the *subroutine*. The bottom line of the main routine says END.

Line 20, which is in the main routine, makes the computer skip to the subroutine. Line 1020, in the subroutine, makes the computer return to the main routine.

To make the program more elegant, change lines 20 and 1020:

```
          10 PRINT "THE"
New line→→20 GOSUB 1000
          30 PRINT "DEAD"
          40 END

          1000 PRINT "BIRD"
          1010 PRINT "IS"
New line→→1020 RETURN
```

The new line 20 says: GO to the SUBroutine that begins at line 1000. The new line 1020 says: RETURN to the main routine, where you left off. Like the old program, the new program prints:

```
THE
BIRD
IS
DEAD
```

GOSUB is like GO TO. The GOSUB 1000 means "GO TO line 1000, and remember where you had come from"; so the computer goes to line 1000, while remembering that it had come from line 20. In line 1020, the RETURN means "RETURN to where you had come from"; so the computer returns to line 20, and then proceeds from line 20 to line 30, which prints the word "DEAD".

The bottom line of a main routine should say END. The bottom line of a subroutine should say RETURN.

Whenever you want the computer to go do a subroutine, say GOSUB.

YANKEE DOODLE

This program prints the *original* version of Yankee Doodle:

```
Print the first verse>>>>>>>>>>>>>>>10 PRINT "FATHER AND I WENT DOWN TO CAMP"
                                  20 PRINT "ALONG WITH CAPTAIN GOODING,"
                                  30 PRINT "AND THERE WE SAW THE MEN AND BOYS"
                                  40 PRINT "AS THICK AS HASTY PUDDING."
Do the chorus, and then come back>>50 GOSUB 1000
Print the second verse>>>>>>>>>>>>>60 PRINT "AND THERE WAS CAPTAIN WASHINGTON"
                                  70 PRINT "UPON A SLAPPING STALLION,"
                                  80 PRINT "A-GIVING ORDERS TO HIS MEN;"
                                  90 PRINT "I GUESS THERE WERE A MILLION!"
Do the chorus, and then come back>>100 GOSUB 1000
Print the third verse>>>>>>>>>>>>>>110 PRINT "THE FLAMING RIBBONS IN HIS HAT,"
                                  120 PRINT "THEY LOOKED SO TARNAL FINE, AH,"
                                  130 PRINT "I WANTED POCKILY TO GET"
                                  140 PRINT "TO GIVE TO MY JEMIMAH."
Do the chorus, and then come back>>150 GOSUB 1000
That's the end of the song>>>>>>>>>160 END
Here's the chorus>>>>>>>>>>>>>>>>>>1000 PRINT
                                  1010 PRINT "YANKEE DOODLE, KEEP IT UP."
                                  1020 PRINT "YANKEE DOODLE DANDY,"
                                  1030 PRINT "MIND THE MUSIC AND THE STEP,"
                                  1040 PRINT "AND WITH THE GIRLS BE HANDY."
                                  1050 PRINT
                                  1060 RETURN
```

Lines 10-40 print the first verse; lines 60-90 print the second; lines 110-140 print the third. At the end of each verse, the computer is told to "GO SUB 1000", which means go to line 1000, print the chorus (lines 1000-1050), and then RETURN to where it left off. Lines 1000-1060, which print the chorus, are the subroutine; the verses (lines 10-160) are the main routine. Lines 1000 and 1050 make the computer print a blank line at the beginning and end of the chorus. The bottom line of the main routine is END; the bottom line of the subroutine is RETURN.

If you accidentally omit line 160 (which says END), the computer will proceed to do the subroutine an extra time, and then won't know where to RETURN. It will gripe, by saying:

```
RETURN BEFORE GOSUB
```

To avoid such gripes, the last line of a main routine should be END.

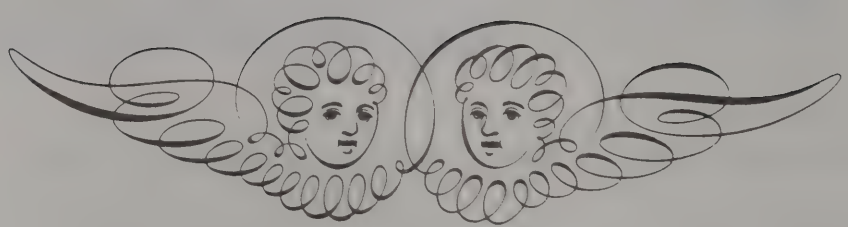
LOVE POEM

This program prints a love poem:

```
10 PRINT "THE MOST BEAUTIFUL THING"
20 PRINT "IN THE WHOLE WIDE WORLD"
30 PRINT "IS..."
40 PRINT "LOVE!"
50 PRINT "THE OPPOSITE OF HATE IS"
60 PRINT "LOVE!"
70 PRINT "THE OPPOSITE OF WAR IS"
80 PRINT "LOVE!"
90 PRINT "THE OPPOSITE OF DESPAIR IS"
100 PRINT "LOVE!"
110 PRINT "AND WHEN I LOOK AT YOU,"
120 PRINT "I FEEL LOTS OF"
130 PRINT "LOVE!"
```

Lines 40, 60, 80, 100, and 130 print the word LOVE. Let's make those lines print the word LOVE larger, like this:

```
*             *      *      *     * * * * *
*           *   *     *     *     *
*         *       *    *   *      * * *
*           *   *       * *       *
* * * * *     *         *         * * * * *
```

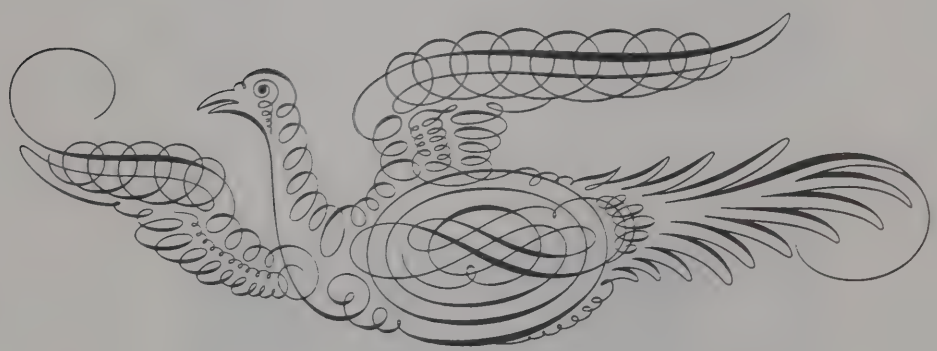


Here's how:

```
          10 PRINT "THE MOST BEAUTIFUL THING"
          20 PRINT "IN THE WHOLE WIDE WORLD"
          30 PRINT "IS..."
New line→→40 GOSUB 1000
          50 PRINT "THE OPPOSITE OF HATE IS"
New line→→60 GOSUB 1000
          70 PRINT "THE OPPOSITE OF WAR IS"
New line→→80 GOSUB 1000
          90 PRINT "THE OPPOSITE OF DESPAIR IS"
New line→→100 GOSUB 1000
          110 PRINT "AND WHEN I LOOK AT YOU,"
          120 PRINT "I FEEL LOTS OF"
New line→→130 GOSUB 1000
New line→→140 END

SUBROUTINE
          →→1000 PRINT "*               *       *       *     * * * * *"
          →→1010 PRINT "*             *   *     *     *     *"
          →→1020 PRINT "*           *       *     *   *       * * *"
          →→1030 PRINT "*             *   *       * *       *"
          →→1040 PRINT "* * * * *       *           *       * * * * *"
          →→1050 PRINT
          →→1060 PRINT
          →→1070 RETURN
```

In that new version, lines 40, 60, 80, 100, and 130 say GOSUB 1000 instead of PRINT "LOVE!". The GOSUB 1000 means: do the subroutine that begins at line 1000. The subroutine prints the word LOVE in large letters.



OLD LADY

Here are the lyrics to a famous song:

There was an old lady who swallowed a fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a bird.
Oh, how absurd! To swallow a bird!
She swallowed the bird to catch the spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a cat.
Imagine that! To swallow a cat!
She swallowed the cat to catch the bird.
She swallowed the bird to catch the spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a dog.
I swear on this log! She swallowed that dog!
She swallowed the dog to catch the cat.
She swallowed the cat to catch the bird.
She swallowed the bird to catch the spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a goat.
Including its coat, she swallowed that goat!
She swallowed the goat to catch the dog.
She swallowed the dog to catch the cat.
She swallowed the cat to catch the bird.
She swallowed the bird to catch the spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a cow.
I don't know how she swallowed that cow!
She swallowed the cow to catch the goat.
She swallowed the goat to catch the dog.
She swallowed the dog to catch the cat.
She swallowed the cat to catch the bird.
She swallowed the bird to catch the spider
That wiggled and jiggled and tickled inside her.
She swallowed the spider to catch the fly,
But I don't know why she swallowed the fly.
Perhaps she'll die.

There was an old lady who swallowed a horse.
And she died, of course!

This program prints the song:

```
10 A$="THERE WAS AN OLD LADY WHO SWALLOWED A "
20 PRINT A$;"FLY.": GOSUB 1070
30 PRINT A$;"SPIDER": GOSUB 1050
40 PRINT A$;"BIRD.": PRINT "OH, HOW ABSURD! TO SWALLOW A BIRD!": GOSUB 1040
50 PRINT A$;"CAT.": PRINT "IMAGINE THAT! TO SWALLOW A CAT!": GOSUB 1030
60 PRINT A$;"DOG.": PRINT "I SWEAR ON THIS LOG! SHE SWALLOWED THAT DOG!": GOSUB 1020
70 PRINT A$;"GOAT.": PRINT "INCLUDING ITS COAT, SHE SWALLOWED THAT GOAT!": GOSUB 1010
80 PRINT A$;"COW.": PRINT "I DON'T KNOW HOW SHE SWALLOWED THAT COW!": GOSUB 1000
90 PRINT A$;"HORSE,": PRINT "AND SHE DIED, OF COURSE!"
100 END

1000 PRINT "SHE SWALLOWED THE COW TO CATCH THE GOAT."
1010 PRINT "SHE SWALLOWED THE GOAT TO CATCH THE DOG."
1020 PRINT "SHE SWALLOWED THE DOG TO CATCH THE CAT."
1030 PRINT "SHE SWALLOWED THE CAT TO CATCH THE BIRD."
1040 PRINT "SHE SWALLOWED THE BIRD TO CATCH THE SPIDER"
1050 PRINT "THAT WIGGLED AND JIGGLED AND TICKLED INSIDE HER."
1060 PRINT "SHE SWALLOWED THE SPIDER TO CATCH THE FLY,"
1070 PRINT "BUT I DON'T KNOW WHY SHE SWALLOWED THE FLY."
1080 PRINT "PERHAPS SHE'LL DIE."
1090 PRINT
1100 RETURN
```

Lines 1000-1100 are a subroutine. The subroutine tells the whole story. In line 20, the GOSUB says to do just *part* of the subroutine—the part that begins at line 1070. Line 30 says to do *more* of the subroutine—by beginning at line 1050. Lines 40, 50, 60, and 70 do progressively larger chunks of the subroutine. Line 80 is the climax: it says to do *all* of the subroutine. Line 90 provides a humorous ending.

NESTING

One subroutine can lead to another:

Main routine→→→→→
```
10 PRINT "MY"
20 GO SUB 1000
30 PRINT "NOSE"
40 END
```

Subroutine 1000→→
```
1000 PRINT "TOE"
1010 PRINT "IS"
1020 GO SUB 2000
1030 PRINT "YOUR"
1040 RETURN
```

Subroutine 2000→→
```
2000 PRINT "STUCK"
2010 PRINT "IN"
2020 RETURN
```

The program consists of three routines. The main routine ends with END; each subroutine ends with RETURN.

The main routine consists of lines 10, 20, 30, and 40. Let's see what those lines do.

Line 10 prints MY.

Line 20 makes the computer do subroutine 1000, which prints TOE, prints IS, does subroutine 2000 (STUCK and IN), and prints YOUR.

Line 30 prints NOSE.

Line 40 ends.

So altogether, the computer prints:

```
MY
TOE
IS
STUCK
IN
YOUR
NOSE
```

In that example, subroutine 2000 is *nested in* subroutine 1000.

ADVANCED MATH

EXPONENTS

The computer understands exponents:

```
PRINT 4↑3
```

That line makes the computer use the number 4, three times. The computer will multiply together those three 4's, like this: 4 times 4 times 4. Since "4 times 4 times 4" is 64, the computer will print:

```
 64
```

The symbols +, -, *, /, and ↑ are all called *operations*.

To solve a problem, the computer uses the three-step process taught in algebra and the "new math". For example, suppose you ask the computer to figure out 70-3↑2+8/2*3. The computer will *not* begin by subtracting 3 from 70; instead, it will use the three-step process:

The problem is . . . . . 70-3↑2+8/2*3

Step 1: get rid of ↑. Now the problem is . . . . . 70- 9 +8/2*3

*Step 2: get rid of * and /. Now the problem is* . . . . . 70- 9 + 12

Step 3: get rid of + and -. The answer is 73

In each step, it looks from left to right. For example, in step 2, it sees / and gets rid of it before it sees *.

Although exponents are fun, the computer handles them slowly. For example, the computer handles 3↑2 more slowly than it handles 3*3. So for speedy calculations, say 3*3 instead of 3↑2.

CONTRASTS

The computer's notation resembles that of arithmetic and algebra, but beware of these contrasts. . . .

For multiplication, the computer requires an asterisk:

Traditional notation	Computer notation
$2n$	`2*N`
$5(n+m)$	`5*(N+M)`
nm	`N*M`

To write a fraction, you must put the numerator in parentheses, if it contains addition or subtraction:

Traditional notation	Computer notation
$\frac{a+b}{c}$	`(A+B)/C`
$\frac{J-20}{6}$	`(J-20)/6`

To write a fraction, put the denominator in parentheses, if it contains addition, subtraction, multiplication, or division:

Traditional notation	Computer notation
$\frac{5}{3+x}$	`5/(3+X)`
$\frac{5a^3}{4b}$	`5*A↑3/(4*B)`

Put the exponent in parentheses, if it contains an operation:

Traditional notation	Computer notation
x^{n+2}	`X↑(N+2)`
x^{3n}	`X↑(3*N)`
$5^{2/3}$	`5↑(2/3)`
2^{3^4}	`2↑(3↑4)`

A *mixed number* is a number that contains a fraction. For example, $9\frac{1}{2}$ is a mixed number. When you write a mixed number, put a plus sign before its fraction:

Traditional notation	Computer notation
$9\frac{1}{2}$	`9+1/2`

If you're using the mixed number in a further calculation, put the mixed number in parentheses:

Traditional notation	Computer notation
$7-2\frac{1}{4}$	`7-(2+1/4)`

SQUARE ROOTS

What positive number, when multiplied by itself, gives 9? The answer is 3, because 3 times itself is 9. Because 3 times itself is 9, 3 is called the *square root* of 9.

To make the computer deduce the square root of 9, type this:

```
PRINT SQR(9)
```

The computer will print 3.

When you tell the computer to PRINT SQR(9), make sure you put the parentheses around the 9.

The symbol SQR is called a *function*. Whenever you use a function, you must put parentheses after it.

The symbol SQR is called a *function*. The number in parentheses (9) is called the function's *input* (or *argument* or *parameter*). The answer, which is 3, is called the function's *output* (or *value*).

SQR(9) gives the same answer as 9↑.5. On maxicomputers and minicomputers, SQR(9) gives the answer more *quickly* than 9↑.5. On microcomputers, the time difference is not significant.

CUBE ROOTS

What number, when multiplied by itself and then multiplied by itself *again*, gives 64? The answer is 4, because 4 times 4 times 4 is 64. The answer (4) is called the *cube root* of 64.

Here's how to make the computer find the cube root of 64:

```
PRINT 64↑(1/3)
```

The computer will print 4.

ABSOLUTE VALUE

To find the *absolute value* of a negative number, just omit the number's minus sign. For example, the absolute value of -7 is 7.

The absolute value of a positive number is the number itself. For example, the absolute value of 7 is 7.

To make the computer find the absolute value of -7, type this:

```
PRINT ABS(-7)
```

The computer will print:

```
7
```

Like SQR, ABS is a function: you must put parentheses after the ABS.

This program computes the distance between two numbers:

```
10 PRINT "I WILL FIND THE DISTANCE BETWEEN TWO NUMBERS."
20 INPUT "WHAT'S THE FIRST NUMBER";X
30 INPUT "WHAT'S THE SECOND NUMBER";Y
40 PRINT "THE DISTANCE BETWEEN THOSE NUMBERS IS";ABS(X-Y)
```

For example, if X is 4, and Y is 7, then the distance between those two numbers is ABS(4-7), which is ABS(-3), which is 3. If you reverse those two numbers, so that X is 7 and Y is 4, the distance between them is ABS(7-4), which is ABS(3), which is still 3.

HOW TO ROUND

If you round 17.9 to an integer, what do you get?

If you round 17.9 to the *nearest* integer, you get 18. If you round 17.9 *down* to an integer, you get 17. If you round 17.9 *up* to an integer, you get 18.

To make the computer round 17.9 *down* to an integer, type this:

```
PRINT INT(17.9)
```

The computer will print:

```
 17
```

Notice that INT rounds *down*. INT(17.9) tells the computer to round 17.9 *down* to an integer; the computer gets 17.

Like SQR and ABS, INT is a function: you must put parentheses after the INT.

If you give this command—

```
PRINT INT(-5.2)
```

what number will the computer print? Will it print -5? Or will it print -6 instead? Answer: the computer will print -6, because if today's temperature is -5.2 degrees, and you round the temperature *down*, the temperature becomes colder: -6 degrees. INT rounds *down*. INT(-5.2) is -6.

INT(54) is simply 54.

To explore further the mysteries of rounding, run this program:

```
10 INPUT "WHAT'S YOUR FAVORITE NUMBER";X
20 PRINT INT(X)
30 PRINT -INT(-X)
40 PRINT INT(X+.5)
```

In that program, line 10 asks you to type a number X. Line 20 prints your number rounded *down*; line 30 prints your number rounded *up*; and line 40 prints your number rounded to the *nearest* integer. For example, if you input 17.9, line 20 makes the computer print 17.9 rounded down (which is 17), line 30 makes the computer print 17.9 rounded up (which is 18), and line 40 makes the computer print 17.9 rounded to the nearest integer (which is 18).

Here's the rule: if X is a number, INT(X) rounds X up; -INT(-X) rounds X down; INT(X+.5) rounds X to the nearest integer.

Rounding down and rounding up are useful in the supermarket. . . .

Suppose some items are marked "30¢ each", and you have only two dollars. How many can you buy? Two dollars divided by 30¢ is 6.66667; rounding *down* to an integer, you can buy 6.

Suppose some items are marked "3 for a dollar", and you want to buy just one of them. How much will the supermarket charge you? One dollar divided by 3 is 33.3333¢; rounding *up* to an integer, you will be charged 34¢.

By using INT, you can do fancier kinds of rounding:

*to round X to the nearest thousand, ask for INT(X/1000+.5)*1000*
to round X to the nearest thousandth, ask for INT(X/.001+.5).001*

This program rounds a number, so that it will have just a *few* digits after the decimal point:

```
10 INPUT "WHAT'S YOUR FAVORITE NUMBER";X
20 INPUT "HOW MANY DIGITS WOULD YOU LIKE AFTER ITS DECIMAL POINT";D
30 B=10↑-D
40 PRINT "YOUR NUMBER ROUNDED IS";INT(X/B+.5)*B
```

Here's a sample run:

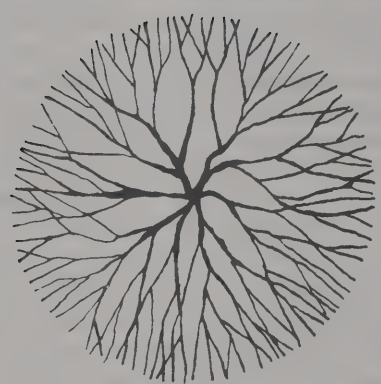
```
WHAT'S YOUR FAVORITE NUMBER? 4.28631
HOW MANY DIGITS WOULD YOU LIKE AFTER ITS DECIMAL POINT? 2
YOUR NUMBER ROUNDED IS 4.29
```

This program rounds a number, so that it will have just a few digits altogether:

```
  10 INPUT "WHAT'S YOUR FAVORITE NUMBER";X
→→20 INPUT "HOW MANY SIGNIFICANT DIGITS WOULD YOU LIKE";D
→→30 B=10↑(1-D+INT(LOG(ABS(X))/LOG(10)))
  40 PRINT "YOUR NUMBER ROUNDED IS";INT(X/B+.5)*B
```

Here's a sample run:

```
WHAT'S YOUR FAVORITE NUMBER? 4.28631
HOW MANY SIGNIFICANT DIGITS WOULD YOU LIKE? 2
YOUR NUMBER ROUNDED IS 4.3
```



STYLE

HOW TO INVENT A PROGRAM

First, decide on your ultimate goal. Be optimistic. Maybe you'd like the computer to play the perfect game of chess? Or translate every English sentence into French?

Research the past

Chances are, whatever you want the computer to do, someone else has thought of the same idea already, and written a program for it.

Find out. Ask your friends. Ask the people in nearby schools, computer stores, computer centers, companies, libraries, and bookstores. Look through books and magazines. There are even books that list what programs have been written. Ask the company you bought your computer from.

Even if you don't find exactly the program you're looking for, you may find one that's close enough to be okay, or that will work with just a little fixing, or can serve as *part* of your program, or will at least give you a *clue* as to where to begin. In one of the textbooks or magazines, you'll probably find a discussion of the problem you're trying to solve, and the pros and cons of various solutions to it—some methods are faster than others.

Remember: if you keep your head in the sand, and don't look at what other people have done already, your programming effort may turn out to be a mere exercise, of no value to the rest of the world.

Simplify

All too often, programmers embark on huge projects and never get them done. Once you have an idea of what's been done before, and how hard your project seems to be, simplify it. Instead of making the computer play a perfect game of chess, how about settling for a game in which the computer plays unremarkably but at least doesn't cheat? Instead of translating every English sentence into French, how about translating just English colors? (We wrote that program already.) In other words, pick a less ambitious, more realistic goal, which if you achieve it, will make you feel good and will be a steppingstone to your ultimate goal.

Finding a bug in a program is like finding a needle in a haystack: it's easier to remove a needle from a small haystack than to wait until more hay has been piled on and then trying to find the same needle.

Specify the I/O

Make your new, simple goal more precise. That's called *specification*. One way to be specific is to draw a picture, showing what your terminal's screen or paper will look like if your program's running successfully.

In that picture, find the lines typed by the computer. They become the PRINT statements in your program. Find the lines typed by the human: they become the INPUT statements. Now you can start writing your program: write the PRINT and INPUT statements on paper, with a pencil, and leave blank lines between them. You'll fill in the blanks later, and won't type on the terminal until your first draft is complete.

Suppose you want the computer to find the average of two numbers. Your picture will look like this:

```
RUN
WHAT'S THE FIRST NUMBER? number
WHAT'S THE SECOND NUMBER? number
THE AVERAGE IS number
```

Your program at this stage will be:

```
10 INPUT "WHAT'S THE FIRST NUMBER";A
20 INPUT "WHAT'S THE SECOND NUMBER";B
etc.
100 PRINT "THE AVERAGE IS";C
```

All you have left to do is figure out what the "etc." is. Here's the general method. . . .

Choose your statements

Suppose you didn't have a computer. Then how would you get the answer?

Would you have to use a mathematical formula? If so, put the formula into your program, but remember that the left side of the equation must have just one variable. For example, if you're trying to solve a problem about right triangles, you might have to use the Pythagorean formula $A^2+B^2=C^2$; but the left side of the equation must have just one variable, so your program must say A=SQR(C↑2-B↑2), or B=SQR(C↑2-A↑2), or C=SQR(A↑2+B↑2), depending on whether you're trying to compute A, B, or C.

Would you have to use a memorized list, such as an English-French dictionary or the population of each state or the weight of each chemical element? If so, that list becomes your DATA, and you need to READ it. If it would be helpful to have the data numbered, so the first piece of data is called X(1), the next piece of data is called X(2), etc., use the DIM statement.

Subscripts are particularly useful if one long list of information will be referred to *several* times in the program.

Does your reasoning repeat? That means your program should have a loop. If you know how many times to repeat, say FOR . . . NEXT. If you're not sure how often, say GO TO. If the thing that's to be repeated isn't repeated immediately, but only after several other things have happened, call the repeated part a *subroutine*, put it at the *end* of your program (followed by RETURN), and say GOSUB whenever you want it done.

At some point in your reasoning, do you have to make a *decision*? Do you have to choose among several alternatives? The way to say "choose" is: IF . . . THEN. If you want the computer to make the choice arbitrarily, "by chance", rather than because of a reason, say: IF RND(2)=1 THEN.

Do you have to compare two things? The way to say "compare A with B" is: IF A=B THEN.

Write pseudocode

Some English teachers say that before you write a paper, you should make an outline. Some computer teachers give similar advice about writing programs.

The "outline" can look like a program in which some of the lines are written in plain English instead of computerese. For example, one statement in your outline might be:

```
130 A = the average of the twelve values of X
```

Such a statement, written in English instead of in computerese, is called *pseudocode*. Later, when you fill in the details, expand that pseudocode into the following:

```
130 S=0
131 FOR I = 1 TO 12
132     S=S+X(I)
133 NEXT
134 A=S/12
```

Organize yourself

Keep the program's over-all organization simple. That will make it easier for you to expand it and find bugs. Here is some folklore, handed down from generation to generation of programmers, that will simplify your organization. . . .

Use top-down programming. That means write a one-sentence description of your program; then expand that sentence to several sentences; then expand each of those sentences to several more sentences; and so on, until you finally expand the sentences that came from sentences that came from sentences into lines of program. Your program will then be in the same order as the English sentences, and therefore will be organized the same way as an English-speaking mind.

A variation is to use subroutines. That means writing the essence of the program as a very short main routine; instead of filling in the grubby details immediately, replace each piece of grubbiness by the word GOSUB. After the main routine is written, write each subroutine. Your program will be like a good book: your main routine will move swiftly, and the annoying details will be relegated to the appendices at the back; the appendices are called *subroutines*. Keep each subroutine down to 50 lines; if it starts getting longer and grubbier, replace each piece of grubbiness by a GOSUB to *another* subroutine, written afterwards and having higher line numbers.

Avoid GO TO. It's hard for a human to understand a program that's a morass of GO TO statements. It's like trying to read a book where each paragraph says to turn to a different page! When you *must* say GO TO, try to go forward instead of backwards, and not go too far.

Divide your program into modules. A *module* is a bunch of consecutive lines forming a unit that cannot be "punctured"; in other words, there is no GO-TO-type statement outside the module that sends the computer to the module's middle; the only way the module can be activated is by starting with its top line. (If the module's particularly nice, the only way it can be deactivated is by arriving at its bottom line; in other words, there is no GO-TO-type statement in the module's middle that sends the computer outside the module.) If you used top-down programming, each module probably corresponds to one sentence in your program's description. Write that sentence at the top of the module, and put an apostrophe to the left of it.

Use variables

After you've written a few lines of your program, you may find that your reasoning "almost repeats"; several lines bear a strong resemblance to each other. You can't use GO TO or FOR . . . NEXT or GOSUB . . . RETURN unless the lines repeat exactly. To make the repetition complete, use a variable to represent the parts that are different.

For example, suppose your program contains these lines:

```
130 PRINT 29.3428+9.87627*SQR(5)
140 PRINT 29.3428+9.87627*SQR(7)
150 PRINT 29.3428+9.87627*SQR(9)
160 PRINT 29.3428+9.87627*SQR(11)
170 PRINT 29.3428+9.87627*SQR(13)
180 PRINT 29.3428+9.87627*SQR(15)
190 PRINT 29.3428+9.87627*SQR(17)
200 PRINT 29.3428+9.87627*SQR(19)
210 PRINT 29.3428+9.87627*SQR(21)
```

Each of those lines says PRINT 29.3428+9.87627*SQR(a number). The number keeps changing, so call it X. Lines 130-210 can be replaced by:

```
130 FOR X= 5 TO 21 STEP 2
140     PRINT 29.3428+9.87627*SQR(X)
150 NEXT
```

Here's a harder example to fix:

```
130 PRINT 29.3428+9.87627*SQR(5)
140 PRINT 29.3428+9.87627*SQR(97.3)
150 PRINT 29.3428+9.87627*SQR(8.62)
160 PRINT 29.3428+9.87627*SQR(.4)
170 PRINT 29.3428+9.87627*SQR(200)
180 PRINT 29.3428+9.87627*SQR(12)
190 PRINT 29.3428+9.87627*SQR(591)
200 PRINT 29.3428+9.87627*SQR(.2)
210 PRINT 29.2428+9.87627*SQR(100076)
```

Again, let's use X. Those nine lines can be combined like this:

```
130 DATA 5,97.3,8.62,.4,200,12,591,.2,100076
140 FOR I = 1 TO 9
150     READ X
160     PRINT 29.3428+9.87627*SQR(X)
170 NEXT
```

This one's even tougher:

```
130 PRINT 29.3428+9.87627*SQR(A)
140 PRINT 29.3428+9.87627*SQR(B)
150 PRINT 29.3428+9.87627*SQR(C)
160 PRINT 29.3428+9.87627*SQR(D)
170 PRINT 29.3428+9.87627*SQR(E)
180 PRINT 29.3428+9.87627*SQR(F)
190 PRINT 29.3428+9.87627*SQR(G)
200 PRINT 29.3428+9.87627*SQR(H)
210 PRINT 29.3428+9.87627*SQR(I)
```

Let's assume A, B, C, D, E, F, G, H, and I have been computed earlier in the program. The trick to shortening those lines is to change the names of the variables. Throughout the program, say X(1) instead of A, say X(2) instead of B, say X(3) instead of C, etc. Say DIM X(9) at the beginning of your program. Then lines 130-210 can be written:

```
130 FOR I=1 TO 9
140     PRINT 29.3428+9.87627*SQR(X(I))
150 NEXT
```

MAKE IT EFFICIENT

Your program should be *efficient*. That means it should use as little of the computer's time and memory as possible.

To use less of the computer's memory, make your DIMensions as small as possible. Try writing the program without any arrays at all; if that turns out to be terribly inconvenient, use the smallest and fewest arrays possible.

To use less of the computer's time, avoid having the computer do the same thing more than once.

These lines force the computer to compute SQR(8.3*N+7) three times:

```
50 PRINT SQR(8.3*N+7)+2
60 PRINT SQR(8.3*N+7)/9.1
70 PRINT 5-SQR(8.3*N+7)
```

You should change them to:

```
→→49 K=SQR(8.3*N+7)
→→50 PRINT K+2
→→60 PRINT K/9.1
→→70 PRINT 5-K
```

These lines force the computer to compute X↑9+2 a hundred times:

```
50 FOR I = 1 TO 100
60     PRINT (X↑9+2)/I
70 NEXT
```

You should change them to:

```
→→49 K=X↑9+2
  50 FOR I = 1 TO 100
→→60     PRINT K/I
  70 NEXT
```

These lines force the computer to count to 100 twice:

```
50 S=0
60 FOR I = 1 TO 100
70     S=S+X(I)
80 NEXT
90 PRINT "THE SUM OF THE X'S IS";S
100 P=1
110 FOR I = 1 TO 100
120    P=P*X(I)
130 NEXT
140 PRINT "THE PRODUCT OF THE X'S IS";P
```

You should change them to:

```
  50 S=0
→→51 P=1
  60 FOR I = 1 TO 100
  70     S=S+X(I)
→→71     P=P*X(I)
  80 NEXT
  90 PRINT "THE SUM OF THE X'S IS";S
  140 PRINT "THE PRODUCT OF THE X'S IS";P
```

Alphabetizing

Suppose you want the computer to alphabetize a list of names. What's the best strategy?

Imagine trying to alphabetize the list yourself—each name is written on a file card, and you have to put the deck of cards in alphabetical order.

One strategy would be to compare the second card with the first, and swap them if necessary; then look at the third card, and swap if necessary; and so on to the end of the deck. A different strategy would be to put all the A's in one pile, all the B's in another, etc., and then sort each pile.

Which strategy is better? If the file has ten cards or less, the swap method is faster; if the file is very long, the 26-pile method is faster but requires space to lay out 26 piles.

Which method would make a more efficient program? That depends on how long the file is and whether your computer's main lack is fast parts or large memory.

Prime numbers

An integer is called *composite* if it's the product of two other integers. 35 is composite, because it is 5*7; 9 is composite, because it's 3*3; 12 is composite, because it's 2*6; 13 is *not* composite, and is therefore called *prime*. This program tells whether a number is prime or composite:

```
10 INPUT "WHAT'S YOUR FAVORITE POSITIVE INTEGER";N
20 FOR I = 1 TO N-1
30     FOR J = 1 TO N-1
40         IF N=I*J THEN PRINT N;"IS";I;"TIMES";J;"AND COMPOSITE": END
50     NEXT
60 NEXT
70 PRINT N;"IS PRIME"
```

Line 10 waits for you to type a number N. Line 40 checks whether N is the product of two other integers; if it is, the computer says N is composite. How efficient is the program? Since it contains no arrays, it doesn't require much space in the memory. But if N turns out to be prime, line 40 is encountered once for every I and once for every J; altogether it's encountered $(N-1)^2$ times. If N is a large number, around a million, $(N-1)^2$ is around a trillion. To do line 40 a trillion times will take a typical microcomputer many years. In fact, if you say that your favorite number is 999983 (which is close to a million), the typical microcomputer will take about *200 years* before it comes to the conclusion that your number is prime! So by the time the program finishes running, you'll be dead, and so will your children! The program is very inefficient.

Some small improvements are possible; for example, I and J can start at 2 instead of 1. But so long as you have a loop inside a loop, the time will remain very large. Here's a strategy that requires only one loop: try dividing N by every integer less than it, and see whether the quotient is ever an integer. Here's the program:

```
 10 INPUT "WHAT'S YOUR FAVORITE POSITIVE NUMBER";N
→→20 FOR I = 2 TO N-1
→→40     Q=N/I: IF Q=INT(Q) THEN PRINT N;"IS";I;"TIMES";Q;"AND COMPOSITE":END
 60 NEXT
 70 PRINT N;"IS PRIME"
```

Line 40 consists of two parts. The first part says to divide N by an integer (I); the quotient's called Q. The second part says that if Q is an integer, N is composite. If N turns out to be prime, line 40 is encountered once for every I; altogether it is encountered N-2 times. That's less than in the previous program, where it was encountered $(N-1)^2$ times. If N is about a million, our new program is nearly a million times faster than the previous one! To determine whether 999983 is prime, the new program takes a typical microcomputer about 3 hours instead of 200 years.

Further improvements are possible. If an N can't be divided by 2, it can't be divided by any even number; so after checking divisibility by 2, we have to check divisibility by just 3, 5, 7, . . . , N-2. Let's put that short-cut into our program, and also say that every N less than 4 is prime:

```
  10 INPUT "WHAT'S YOUR FAVORITE NUMBER";N
→→11 IF N<4 THEN PRINT N;"IS PRIME":END
→→12 Q=N/2: IF Q=INT(Q) THEN PRINT N;"IS 2 TIMES";Q;"AND COMPOSITE": END
→→20 FOR I = 3 TO N-2 STEP 2
  40     Q=N/I: IF Q=INT(Q) THEN PRINT N;"IS";I;"TIMES";Q;"AND COMPOSITE": END
  60 NEXT
  70 PRINT N;"IS PRIME"
```

Line 12 checks divisibility by 2; lines 20-60 check divisibility by 3, 5, 7, . . . , N-2. If N is prime, line 40 is encountered N/2 - 2 times, which is about half as often as in the previous program; so our new program takes about half as long to run. On a typical microcomputer, it takes about 1½ hours to handle 999983.

Our original object was to find a pair of integers whose product is N. If there is such a pair of integers, the smaller one will be no more than $\sqrt{N}$, so we can restrict out hunt to the integers not exceeding $\sqrt{N}$:

```
  10 INPUT "WHAT'S YOUR FAVORITE NUMBER";N
  11 IF N<4 THEN PRINT N;"IS PRIME":END
  12 Q=N/2: IF Q=INT(Q) THEN PRINT N;"IS 2 TIMES";Q;"AND COMPOSITE": END
→→20 FOR I = 3 TO SQR(N)*1.00001 STEP 2
  40     Q=N/I: IF Q=INT(Q) THEN PRINT N;"IS";I;"TIMES";Q;"AND COMPOSITE": END
  60 NEXT
  70 PRINT N;"IS PRIME"
```

The "1.00001" is to give a margin of safety, in case the computer rounds SQR(N) a bit down. If N is near a million, line 40 is encountered about 500 times, which is much less than the 500,000 times encountered in the previous program and the 1,000,000,000,000 times encountered in the original. This program lets the typical microcomputer handle 999983 in about 6 seconds. That's much quicker than the earlier versions, which required 1½ hours, or 3 hours, or 200 years!

Moral: a few small changes in a program can make the computer take 6 seconds instead of 200 years.

The frightening thing about this example is that the first version we had was so terrible, but the only way to significantly improve it was to take a totally fresh approach. To be a successful programmer, you must always keep your mind open, and hunt for fresh ideas.

DON'T BE SILLY

Skim through your program and eliminate any lines that are silly. Here are examples of silly lines; I assume your program is numbered 10, 20, 30, 40, . . .

A GO TO that goes to the next line:

```
30 GO TO 40
```

Omit it. The computer will go to line 40 anyway.

Here's another example of a GO TO that goes to the next line:

```
30 IF X<7 THEN GO TO 40
```

Omit it. The computer will go to line 40 anyway.

An IF that just skips over just one line:

```
30 IF X<7 THEN GO TO 50
40 PRINT "WOW"
```

In that example, line 30 is an IF that says to skip over just line 40. Combine lines 30 and 40 into a single line:

```
30 IF X>=7 THEN PRINT "WOW"
```

The new line 30 says: if X>=7, print WOW; but if X<7, proceed to the next line, which is now line 50.

An IF followed by its opposite:

```
30 IF X<7 THEN GO TO 100
40 IF X>=7 THEN GO TO 200
```

Remove the IF from line 40:

```
  30 IF X<7 THEN GO TO 100
→→40 GO TO 200
```

The new version does the same thing as the original, because the computer reaches the new 40 only if X is not less than 7.

Here's another IF followed by its opposites:

```
30 IF X<7 THEN GO TO 100
40 IF X=7 THEN GO TO 200
50 IF X>7 THEN GO TO 300
```

Remove the last IF:

```
  30 IF X<7 THEN GO TO 100
  40 IF X=7 THEN GO TO 200
→→50 GO TO 300
```

By removing one piece of silliness, you may uncover another. For example, here's a piece of silliness—an IF followed by its opposite:

```
30 IF X<7 THEN GO TO 50
40 IF X>=7 THEN GO TO 200
```

Applying our rules of cosmetics, we get:

```
  30 IF X<7 THEN GO TO 50
→→40 GO TO 200
```

But now line 30 is an IF that skips over just one line; we've uncovered another piece of silliness! Applying more cosmetics, we get down to a single line:

```
30 IF X>=7 THEN GO TO 200
```

AVOID ROUND-OFF

The computer cannot handle decimals accurately. If you say X=.1, the computer can't set X equal to .1 exactly; instead, it will set X equal to a number very, very close to .1. The reason for the slight inaccuracy is that the computer thinks in "binary", not decimals; and .1 can*not* be expressed in binary exactly.

Usually you won't see the slight inaccuracy: when you ask the computer to PRINT a number, the computer prints it rounded to six significant figures, and the inaccuracy is so small it doesn't show up in the rounded result. But there are three situations in which the inaccuracy can be noticed:

1. Telling the computer to do A-B, where A is almost equal to B, and the first several digits of A are the same as the first several digits of B. For example, if you ask a typical microcomputer to print 8.001-8, the computer will *not* print .001; instead it will print .000999451. The same thing happens if you do 8.001+(-8).

If you ask the typical microcomputer to print 987654.1-987654, it will print .125 instead of .1. The error can get magnified: if you ask the computer to multiply 987654.1-987654 by 1000, it will print .125*1000, which is 125, instead of .1*1000, which is 100. If you ask it to find the reciprocal of 987654.1-987654, it will print 1/.125, which is 8, instead of 1/.1, which is 10.

Those are the errors produced by a *typical* microcomputer. The errors produced by *your* computer might be slightly more or less. But even if your computer is a maxi that costs $10,000,000, it makes those same kinds of errors.

2. Saying "FOR X = A TO B STEP C", where C is a decimal and the loop will be done many times. For example:

```
10 FOR X = 1 TO 2 STEP .1
20     PRINT X
30 NEXT
```

Theoretically, the computer should print 1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, and 2. But that's not what actually happens. In line 10, the computer can't handle the decimal .1 accurately. The last few numbers the typical microcomputer thinks of are:

slightly more than 1.7
slightly more than 1.8
slightly more than 1.9

The computer does *not* think of the next number, slightly more than 2.0, because line 10 says not to go past 2. In line 20, the word PRINT makes the computer print the numbers rounded to six significant digits, so it prints:

```
 1
 1.1
 1.2
 1.3
 1.4
 1.5
 1.6
 1.7
 1.8
 1.9
```

It does not print 2.

If you want to compute 1 + 1.1 + 1.2 + 1.3 + 1.4 + 1.5 + 1.6 + 1.7 + 1.8 + 1.9 + 2, you might be tempted to write this program:

```
→→5 S=0
  10 FOR X = 7 TO 193 STEP .1
→→20     S=S+X
  30 NEXT
→→40 PRINT S
```

The computer will print a reasonable-looking answer: 14.5. But that "answer" is wrong, because the last number it added was slightly more than 1.9; it never added 2. The correct answer is 16.5.

One remedy is to change line 10 to this:

```
10 FOR X = 1 TO 2.05 STEP .1
```

The .05 after the 2 allows for the margin of error. The general strategy is to change—

```
10 FOR X = A TO B STEP C
```

to this:

```
10 FOR X = A TO B+C/2 STEP C
```

An alternative remedy is to replace—

```
10 FOR X = 1 TO 2 STEP .1
```

by this pair of lines:

```
10 FOR I = 10 TO 20
11     X= I/10
```

As I goes from 10 to 20, X will go from 1 to 2 in steps of .1. This remedy is the most accurate of all, since it eliminates decimals from line 10. But the division in line 11 makes the program very slow.

3. Asking the computer whether two numbers X and Y are equal. It's unwise to ask whether X is *exactly* equal to Y, since both X and Y have probably been affected by some slight error. Instead, ask the computer whether the difference between X and Y is much tinier than Y:

Bad	Good
`IF X=Y THEN`	`IF ABS(X-Y)<=.000001*ABS(Y) THEN`

The .000001 is requesting, roughly, that the first six significant digits of X be the same as the first six significant digits of Y.

Why binary?

From those discussions, you might think computers should be made differently, and that they should use the decimal system instead of binary. There are two counterarguments.

First, binary arithmetic is faster.

Second, even if computers were using the decimal system, inaccuracy would still occur. To store the fraction 2/3 accurately by using the decimal system, the computer would have to store a decimal point followed by infinitely many 6's. That would require an infinite amount of space in memory, which is impossible (unless you know how to build an infinitely large computer?) So even in the decimal system, some fractions must be approximated instead of handled exactly.

Begin with the tiny

According to mathematicians, addition is supposed to obey these laws:

A+0 is exactly the same as A
A+B is exactly the same as B+A
A+-A is exactly the same as 0
(A+B)+C is exactly the same as A+(B+C)

On the computer, the first three laws hold, but the last does not. If A is a decimal tinier than C, the computer does (A+B)+C more accurately than A+(B+C). So to add a list of numbers accurately, begin by adding together the tiniest decimals in the list.

TEST YOUR PROGRAM

When you've written a program, *test* it: type RUN and see whether it works.

If the computer does *not* gripe, your tendency will be to say "Whoopee!" Don't cheer too loudly. The answers the computer is printing may be wrong. Even if its answers look reasonable, don't assume they're right: the computer's errors can be very subtle. Check its answers, by doing some calculations on a piece of paper.

Even if the answers the computer prints are correct, don't cheer. Maybe you were just lucky. Type different input, and see whether your program still works. Chances are, there's something you can input that will make your program go crazy or print a wrong answer. Your mission: to find input that will reveal the existence of a bug. Try six kinds of input. . . .

1. Simple input. Type in simple integers, like 2 and 10, so the computation is simple, and you can check the computer's answers easily.

2. Increasing input. See how the computer's answer changes when the input changes from 2 to 1000. Does the change in the computer's answer look reasonable? Does the computer's answer go up when it should go up, and down when it should go down? and by a reasonable amount?

3. Input that test each IF. For a program that says—

```
30 IF X<7 THEN GO TO 100
```

input an X less than 7 (to see whether line 100 works), input an X greater than 7 (to see whether line 40 works), input an X equal to 7 (to see whether you really want "<" instead of "<="), and input an X very close to 7, to check round-off error. For a program that says—

```
30 IF A↑2+B<C THEN GO TO 100
```

input an A, B, and C that make A↑2+B less than C; that make A↑2+B close to C.

4. Extreme input. What happens if you input:

a huge number, like 45392000000 or 1E35?
a tiny number, like .00000003954 or 1E-35?
a trivial number, like 0 or 1?
a typical number, like 45.13?
a negative number, like -52?

Find out. If the input is supposed to be a string, what happens if you input AAAAA or ZZZZZ? If there are supposed to be two inputs, what happens if you input the same thing for each?

5. Input that will make some lines in your program act strangely. If your program contains division, create input that will make the divisor be zero or a tiny decimal close to zero. If your program contains a square root of quantity, create input that will make the quantity be negative. If your program says "FOR I= A TO B", create input that will make B be less than A, or equal to A. If your program mentions X(I), create input that will make I be zero or negative or greater than the DIM. Create input that causes round-off error: for a program that says "A-B" or says "IF A=B THEN", create input that will make A almost equal B.

6. Garbage. Many people hate computers because they often print wrong answers. A computer can print a wrong answer because its machinery is broken, or because a program has a bug. But the main reason why computers print wrong answers is that the input is incorrect. Incorrect input is called *garbage*. It has several causes. . . .

The user's finger slips. Instead of 400, he inputs 4000. Instead of 27, he inputs 72. Trying to type .753, he leaves out the decimal point.

The user got wrong information. He tries to input the temperature, but his thermometer is leaking. He tries to input the results of a questionnaire, but everybody who filled out his questionnaire lied.

The instructions aren't clear. The program asks HOW FAR DID THE BALL FALL, and the user doesn't know whether to type the distance in feet or in meters. Is time to be given in seconds or minutes? Are angles to be measured in degrees or radians? If the program asks WHAT IS YOUR NAME, should the user type JOE SMITH or "SMITH,JOE" or just JOE? Can the user input Y instead of YES? Maybe the user isn't clear about whether to insert commas, quotation marks, and periods. If several items are to be typed, should they be typed on the same line or on separate lines? If your program asks HOW MANY BROTHERS AND SISTERS DO YOU HAVE, and the user has 2 brothers and 3 sisters, should he type 5 or "2,3" or "2 BROTHERS AND 3 SISTERS"? For a quiz that asks WHO WAS THE FIRST PRESIDENT OF THE UNITED STATES, what if the user answers GEORGE WASHINGTON or simply WASHINGTON or G. WASHINGTON or GENERAL GEORGE WASHINGTON or PRESIDENT WASHINGTON or MARTHA'S HUSBAND? Make the instructions clearer:

```
WHO WAS THE FIRST PRESIDENT OF THE UNITED STATES (GIVE JUST HIS LAST NAME)?
```

The user is trying to be funny or sabotage the computer. Instead of inputting his name, he types an obscene comment. When asked how many brothers and sisters he has, he says 275.

It's your responsibility as a programmer to make sure that the directions for using your program are clear, and that the program rejects ridiculous input. For example, if your program is supposed to print weekly paychecks, it should refuse to print checks for more than $10000. Your program should contain these lines:

```
20 INPUT "HOW MUCH MONEY DID THE EMPLOYEE EARN";E
30 IF E>10000 THEN PRINT E;"IS QUITE A BIG PAYCHECK!  I DON'T BELIEVE YOU": PRIN
T "PLEASE RETYPE YOUR REQUEST.": GO TO 20
```

Line 30 is called an *error trap* (or an *error-handling routine*). Your program should contain several, to prevent printing checks that are too small (2¢?) or negative or otherwise ridiculous ($200.73145?)

To see how your program reacts to input that's either garbage or unusual, ask the person sitting next to you to run your program. That person might input something you never thought of.

DOCUMENT IT

Write an explanation that helps other people understand your program. An explanation is called *documentation*; when you write an explanation, you're *documenting* the program.

You can write the documentation on a separate sheet of paper (to be put in the computer center's library), or you can make the computer print the documentation when the user types RUN or LIST.

A popular device is to begin the RUN by making the computer ask the user:

```
DO YOU NEED INSTRUCTIONS?
```

You need two kinds of documentation: how to use the program, and how the program was written.

Your explanation of how to use the program should include:

the program's name

how to get the program from the disk

the program's purpose

a list of other programs that must be combined with this program, to make a workable combination

the correct way to type the input and data (show an example)

the correct way to interpret the output

the program's limitations (input it can't handle, a list of error messages that might be printed, round-off error)

a list of bugs you haven't fixed yet

An explanation of how you wrote the program will help other programmers borrow your ideas, and help them expand your program to meet new situations. It should include:

your name

the date you finished it

the computer you wrote it for

the language you wrote it in (probably BASIC)

the name of the method you used ("This program solves quadratic equations by using the quadratic formula, solves triangles by using the law of sines, solves simultaneous linear equations by using Gauss-Jordan elimination, and alphabetizes the data by using a bubble sort.")

the name of the book or magazine where you found the method

the name of any program you borrowed ideas from

*an informal explanation of how the program works ("It keeps looping until A↑2 is greater than 2*B, which makes it jump to subroutine 1000 and compute the weather forecast for Tuesday.")*

the purpose of each module

the meaning of each variable

the meaning of arriving at a line (for a program saying IF X<60 THEN GO TO 1000, you might say, "Arrival at line 1000 indicates the student flunked.")

CHAPTER 7

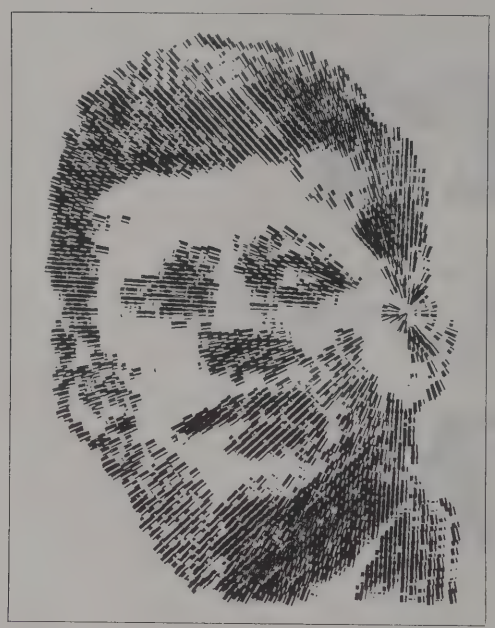
ARTS & BRAINS

GRAY LEVELS

You can express every black-and-white photograph as a table of numbers. Each number in the table represents the darkness of a different point— the higher the number, the darker the point. The "darkness numbers" are called *gray levels*. To feed a picture into the computer, type in the table of gray levels. Or aim a special camera at the object you want pictured; the camera system will automatically compute the gray levels and send them to the computer via a wire.

You can program the computer to change the gray levels in any weird way you wish, and draw the result. Back in the 1960's, the Computer Technique Group of Japan did this to an ordinary photograph of John Kennedy:

Shot Kennedy

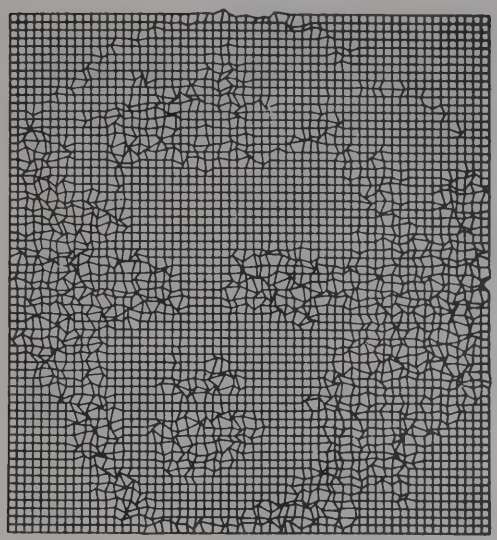


Diffused Kennedy

Kennedy in a Dog

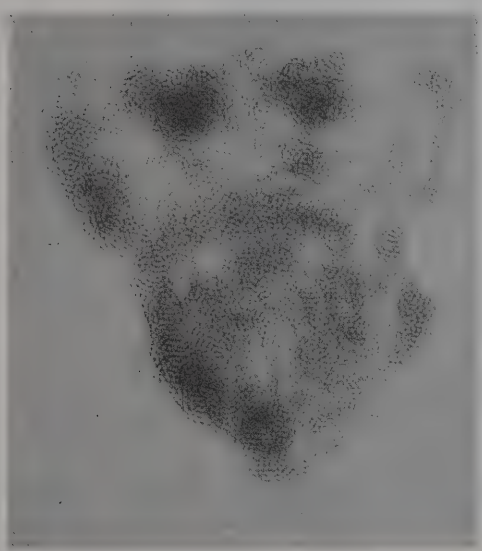
Here's what the group did to a photograph of Marilyn Monroe:

Monroe in the Net



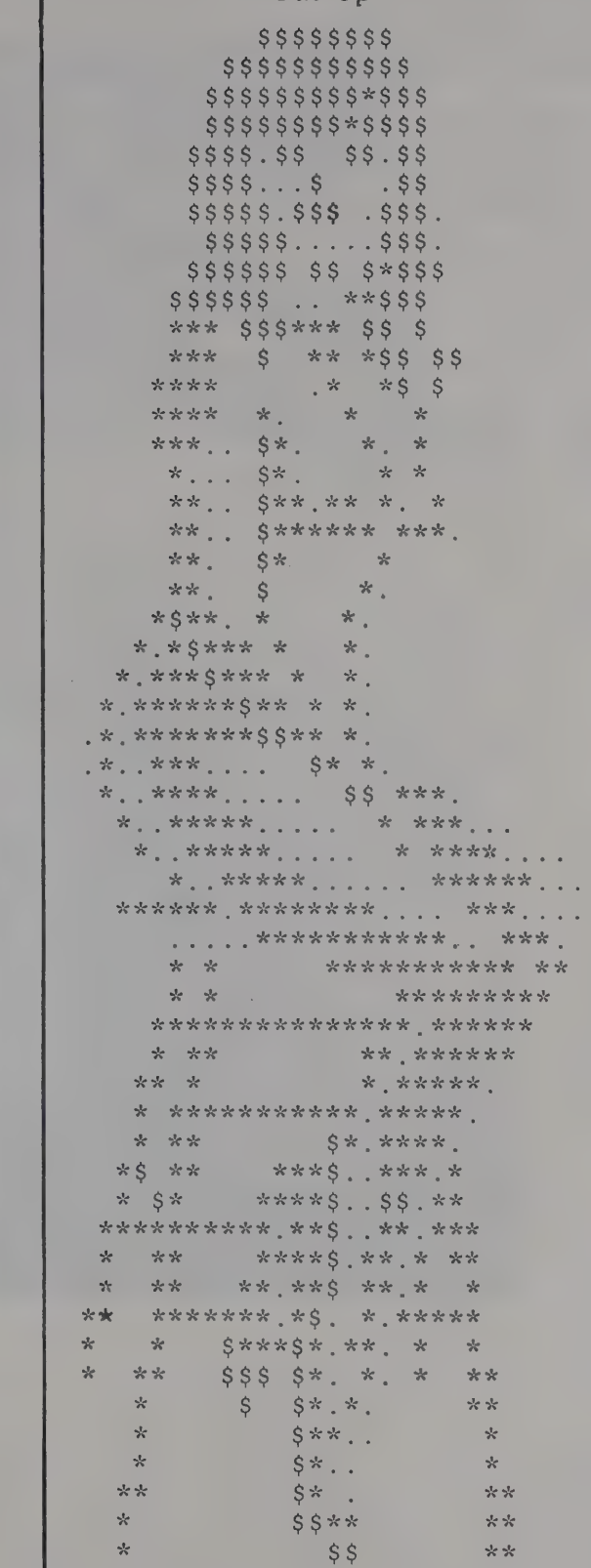
Csuri & Shaffer fed the computer a realistic line drawing of an old man; here's what came out:

Random Light and Shadow



I did this with the help of a computer:

Pin-Up



The *Pin-Up* has these specifications....

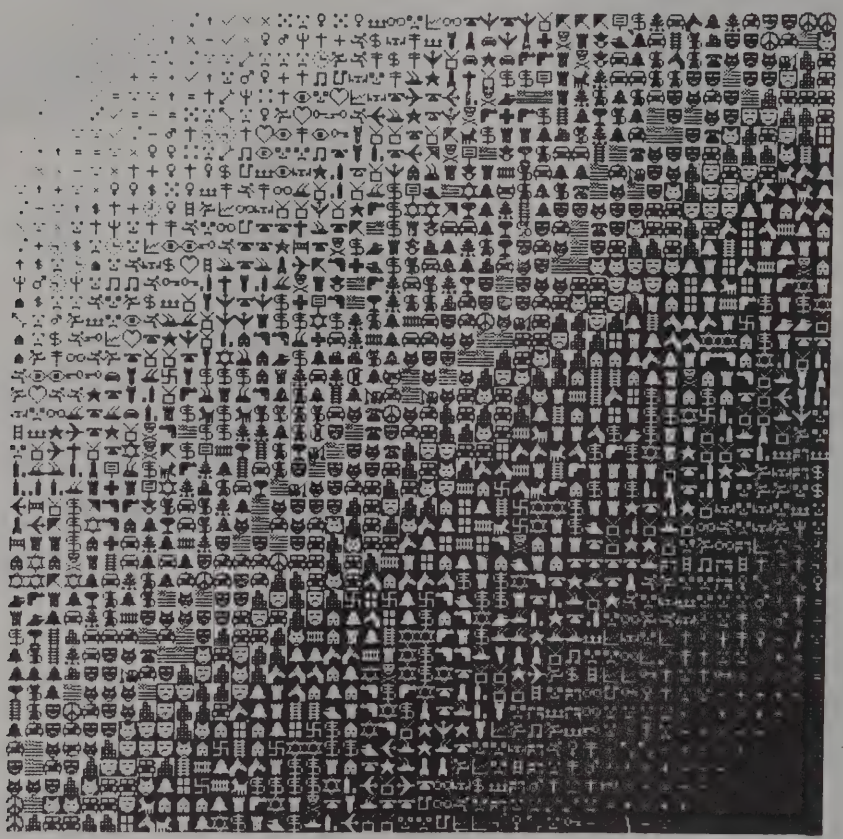
scene: a scantily clad girl sitting on a stool
4 different gray levels
4 different symbols (1 for each gray level): a blank, a period, an asterisk, and $
1537 symbols altogether (53 rows × 29 columns)

In the specification, the numbers are small, yet the picture is clear. To obtain the clarity, I did non-computerized fanagling.

At Bell Telephone Laboratories, Knowlton & Harmon produced a picture with much larger specifications....

scene: two seagulls flying in the clouds
16 different gray levels
141 different symbols (for each gray level there are several symbols; the computer chooses among them at random)
11616 symbols altogether (88 rows × 132 columns)

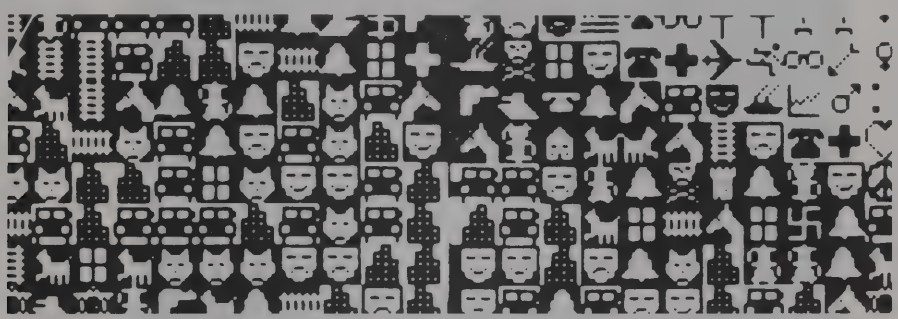
Instead of using blanks, periods, asterisks, and $, they used cats, battleships, swastikas, and other weird shapes. Here are the 141, listed from lightest to darkest, with some repetitions:



The picture is several feet long. Seen from a distance, it looks like this:

Gulls

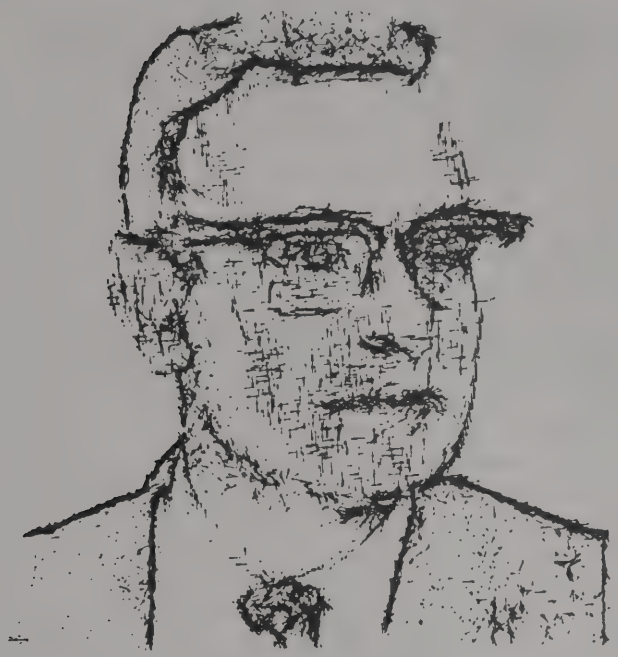
Here's a close-up view of part of one of the gull's wings:



If you don't like seagulls, how about Mona Lisa?

In 1971, Michael Hord made the computer turn photographs into artistic sketches. Here's what the computer did to a photograph of his boss, and to a photograph of a colleague's girlfriend:

Boss



Woman

To draw each sketch, the computer's camera scanned the original photograph, and found the points where the photograph changed dramatically from light to dark. Then, on a sheet of paper, it plotted those points; and through each of those points, it drew a short line perpendicular to the direction in which the original photograph darkened.

More precisely, here's what the computer did....It looked at four adjacent points on the original photograph:

A B
C D

It computed the darkness of each of those points. Then it computed the "darkening in the X direction", defined as:

(the darkness at B) + (the darkness at D) - (the darkness at A) - (the darkness at C)

Then it computed the "darkening in the Y direction", defined as:

(the darkness at A) + (the darkness at B) - (the darkness at C) - (the darkness at D)

Then it computed the "overall darkening", defined as:

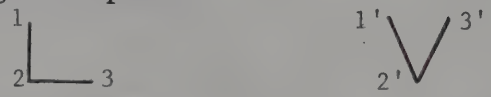
$$(\text{the darkening in the X direction})^2 + (\text{the darkening in the Y direction})^2$$

If the overall darkening there turned out to be large, the computer sketched a short line, in the vicinity of the points ABCD, and perpendicular to the direction of darkening. More precisely, the line's length was 1, and the line's slope was:

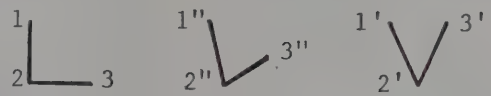
$$-\frac{\text{the darkening in the X direction}}{\text{the darkening in the Y direction}}$$

TRANSFORMATION

Here's how to make an L slowly become a V. Notice that the letters L and V are both made by connecting three points:



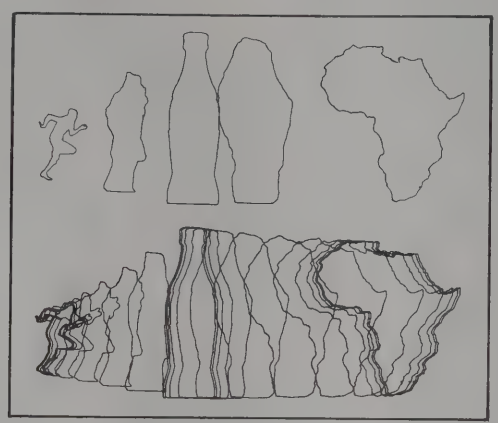
Let 1" be the point halfway between 1 and 1'; let 2" be halfway between 2 and 2'; and let 3" be halfway between 3 and 3'. Then 1", 2", and 3" form a shape that's halfway between an L and a V:



The process can be extended further:

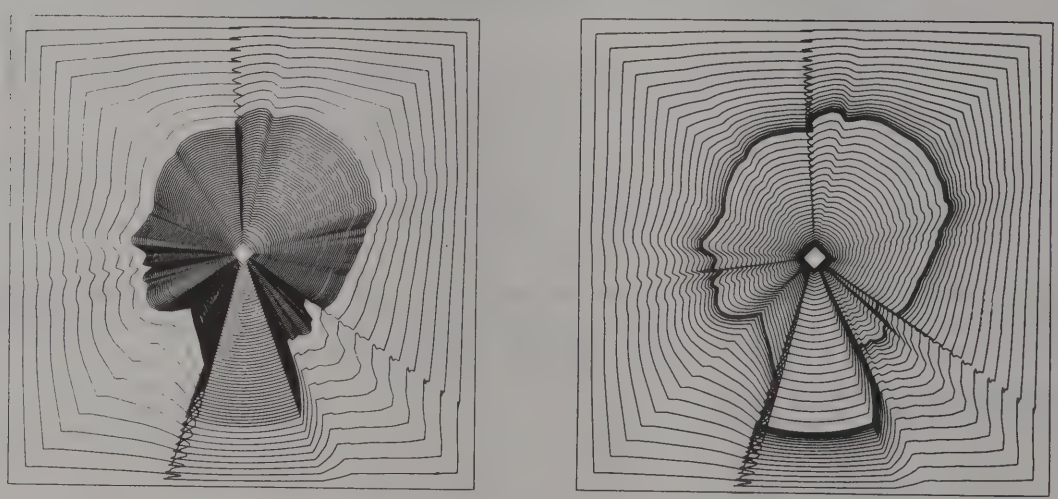
Using that method, the Computer Technique Group of Japan gradually turned a running man into a Coke bottle, and then into Africa:

Running Cola is Africa



The group turned this head into a square:

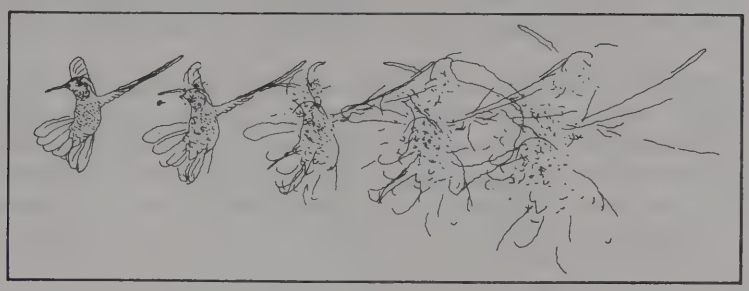
Return to a Square



The head on the left returns to a square by using *arithmetic progression*: the lines are equally spaced. The one on the right uses *geometric progression* instead: the lines are close together near the inside square, but further apart as they expand outward.

Csuri & Shaffer exploded a hummingbird:

Chaos to Order



The hummingbird at the far right was obtained from the one at the far left, by moving each line a random distance and in a random direction (between 45° and -45°).

Computers can make movies. The best movie ever made by a computer is called *Hunger* (or *La Faim*). It was made under the auspices of the Canadian Film Board. It's a 10-minute cartoon, in color, with music; but it goes far beyond anything ever done by Walt Disney. It uses the same technique as *Running Cola is Africa*: it shows objects turning into other objects. It begins by showing a harried, thin executive at his desk, which has two phones. One of the phones rings. He answers it. While he's talking on that phone, his other phone rings. To talk on both phones simultaneously, his body splits in two. (How does a single body become two bodies? By using the same technique as turning a running man into a coke bottle.) On the other side of his desk is an armchair, which turns into a secretary, whose head turns into a clock saying 5PM, which tells the executive to go home. So he stretches his arms in front of him, and becomes his car: his hands become the headlights, his arms become the front fenders, his face becomes the windshield: you have to see it to believe it! He drives to a restaurant and gets the waitress, who turns into an ice-cream cone; then he eats her. As the film progresses, he becomes increasingly fat, lustful, slothful, and miserable. In the end, he falls into hell, where he's encircled by all the poor starving naked children of the world, who eat his flesh. Then the film ends. (Don't see it before eating dinner!). It combines computer art and left-wing humanitarian politics, to create an unforgettable message.

ORDER VERSUS DISORDER

Computer artists are starting to believe that art is a tension between order and disorder. Too much order, or too much disorder, will bore you. For example, in *Chaos to Order*, the hummingbird on the left is too orderly to be art. The hummingbird on the right is more interesting.

Consider *Gulls* (page 37). Seen from a distance, it's an orderly picture of gulls. Seen up close, it's an orderly picture of a cat or a battleship or a swastika. But from a middling distance, it looks like disorderly wallpaper— the symbols repeat, but not in any obvious cycle. That element or disorder is what makes the picture interesting.

At first glance, *Pin-Up* (page 35) is just a disorderly array of periods, asterisks, and dollar signs. At second glance, you see order: a girl. Art is the formation of order from disorder.

A first glance at *Monroe in the Net* (page 35) shows order: a piece of graph paper. A second glance shows disorder: some of the graph's lines are inexplicably bent. A third glance shows order: Marilyn Monroe's face pops out at you. Her orderly face is formed from the disorder or bent lines.

Return to a Square (page 40) uses arithmetic progression and geometric progression to create an over-all sense of order, but the basic elements are *dis*orderly: a head that's bumpy, and a panorama of weird shapes that lie uncomfortably between being heads and squares but are neither.

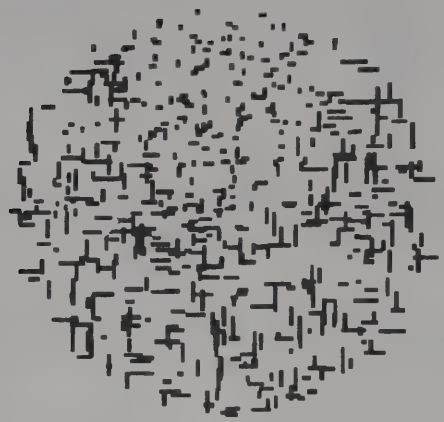
Many programs create disorder by random numbers. *Chaos to Order* uses random numbers to explode the hummingbird. *Gulls* uses random numbers to help choose among the 141 symbols.

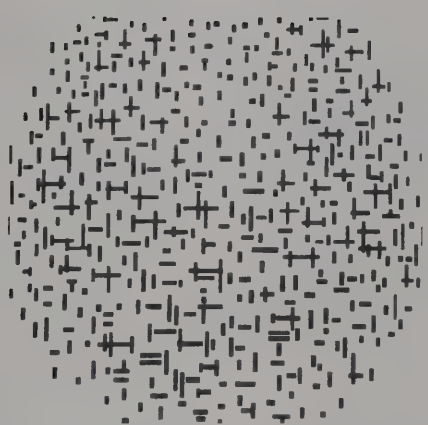
An amazing example of random numbers is this picture by Julesz & Bosche:

To your eyes, the picture seems quite ordered. Actually, it's quite *dis*ordered. One pie-shaped eighth of it is entirely random; the other seven eighths are copies of it. The copying is the only element of order, but very powerful. Try this experiment: *cover seven-eighths of the picture*. You'll see that the remaining eighth is totally disordered, hence boring.

That program imitates a child's *kaleidoscope*. Do you remember your childhood days, when you played with your kaleidoscope? It was a cardboard "telescope" that contained a disorganized pile of colored glass and stones, plus a series of mirrors that produced eight-way symmetry, so that what you saw resembled a giant multicolored snowflake. The program by Julesz & Bosche uses the same technique, computerized. Hundreds of programmers have imitated Julesz & Bosche, so that today you can buy kaleidoscope programs for your Apple, Radio Shack, etc. Or try writing your own!

Take this test:





One of those is a famous painting (*Composition with Lines*, by Piet Mondrian, 1917). The other was done by a computer (programmed by A. Michael Noll in 1965). *Which one was done by the computer? Which one do you like best?*

The solution is on the next page, but *don't peek until you've answered!*

The computer did the top one.

The programmer surveyed 100 people. Most of them (59) thought the computer did the bottom one. Most of them (72) preferred the top one— the one that was actually done by the computer.

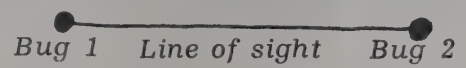
The test shows that people can't distinguish computer art from human art, and that the computer's art is more pleasing than the art of a famous painter.

The computer's version is more disordered than Mondrian's. The computer created the disorder by using random numbers. The survey shows that most people like disorder: Mondrian's work is too ordered. It also shows that most people mistakenly think the "computer" means "order".

ENVELOPES

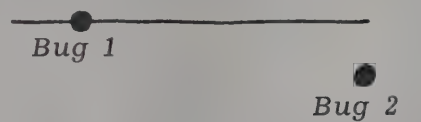
Try this experiment. On a piece of paper, put two dots, like this:

The dots represent little insects, or "bugs". The first bug is looking at the second bug. Draw the first bug's line of sight:

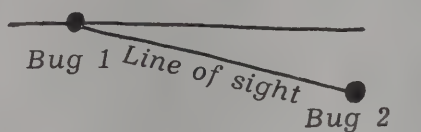


Make the first bug take a step toward the second bug:

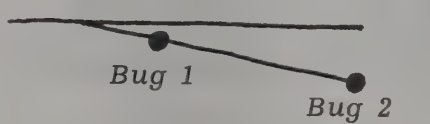
Make the second bug run away, in any direction:



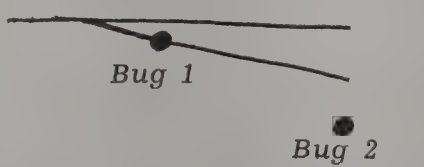
Now repeat the entire process. Again, bug 1 looks at bug 2; draw its line of sight:



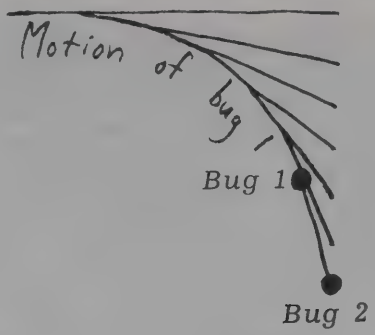
Bug 1 moves toward bug 2:



Bug 2 keeps running away:



If you repeat the process many times, you get this:



The "motion of bug 1" looks like a curve. (In fact, it's a parabola.) The "curve" is composed of many straight lines— the lines of sight. That's how to draw a fancy curve by using straight lines.

Each straight line is called a *tangent* of the curve. The entire collection of straight lines is called the curve's *envelope*. Creating a curve, by drawing the curve's envelope, is called *stitching the curve*— because the lines of sight act as threads, to produce a beautiful curved fabric.

You can program the computer to draw those straight lines. That's how to make the computer draw a fancy curve— even if you know nothing about "equations of curves".

To get a curve that's more interesting, try these experiments:

What if bug 2 doesn't walk in a straight line? What if bug 2 walks in a curve instead?

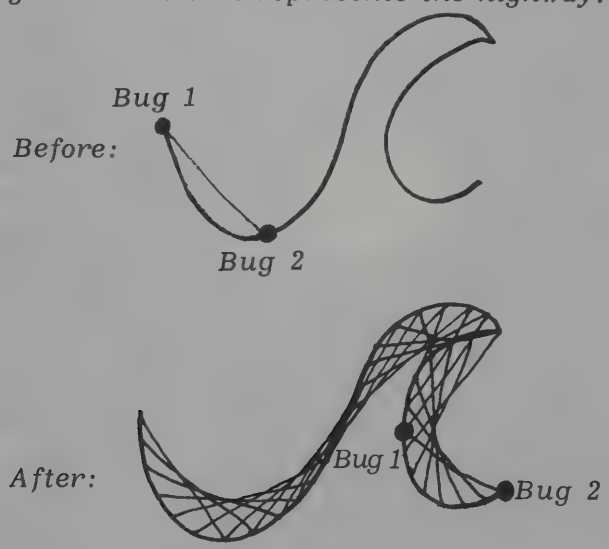
What if bug 1 goes slower than bug 2, and takes smaller steps?

What if the bugs accelerate, or slow down?

What if there are THREE bugs? What if bug 1 chases bug 2, while bug 2 chases bug 3, while bug 3 chases bug 1?

What if there are MANY bugs, all chasing each other, and their starting positions are random?

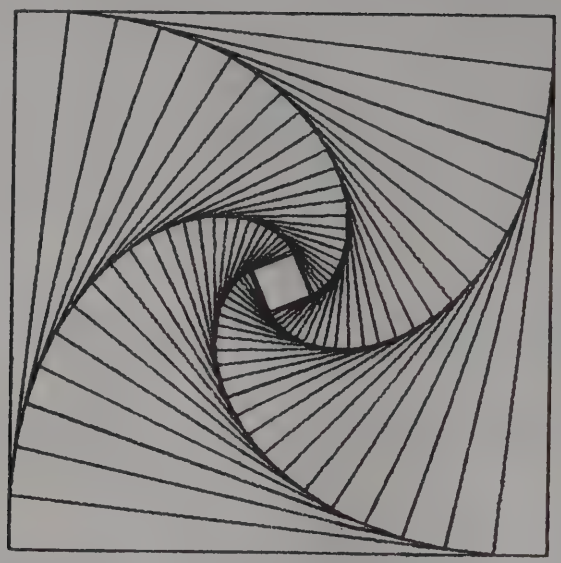
What if there are just two bugs, but the bugs are Volkswagens, which must drive on a highway having nasty curves? Show the bugs driving on the curved highway. (Their lines of sight are still straight; but instead of moving along their lines of sight, they must move along the curve that represents the highway.)



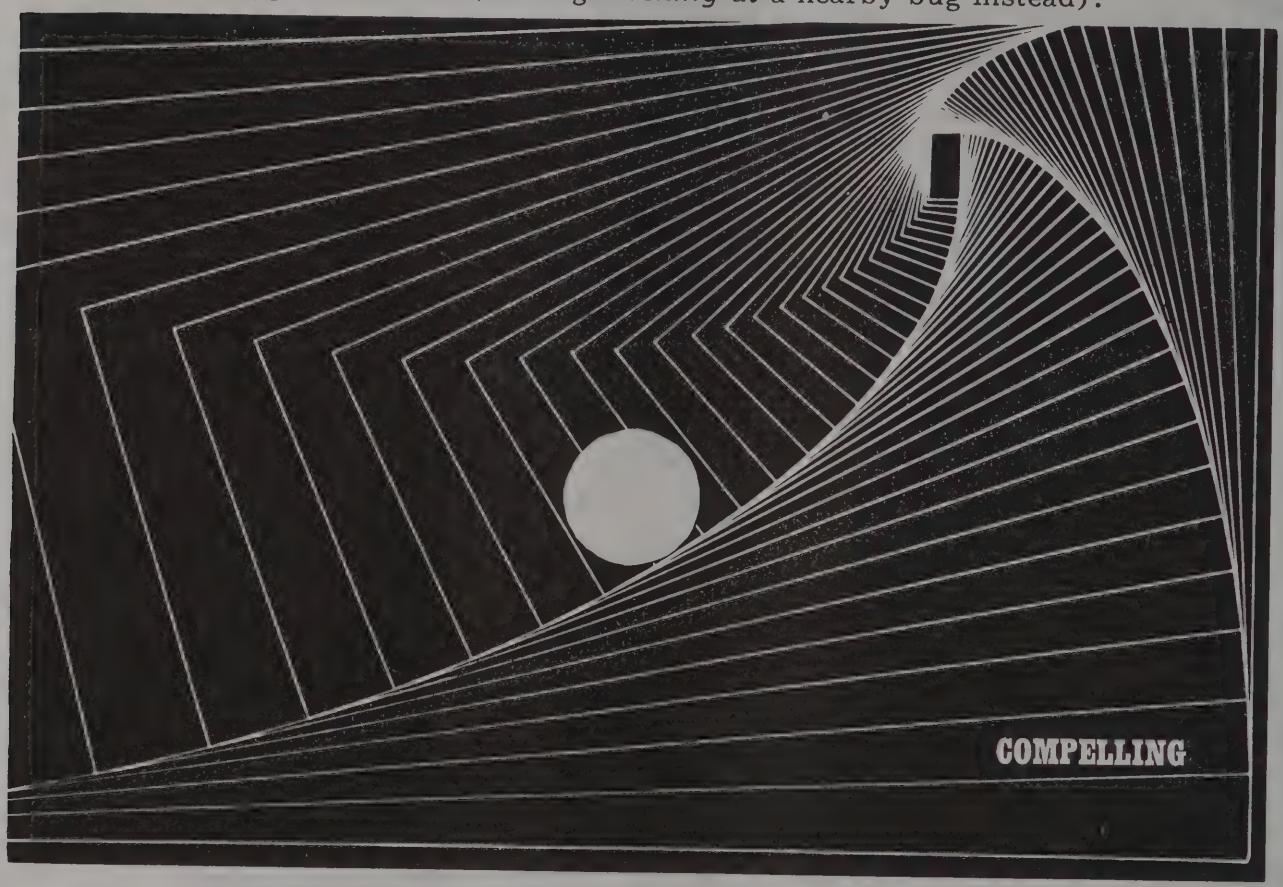
What if each bug has his own highway, and all the bugs stare at each other?

Here are some elaborate examples.

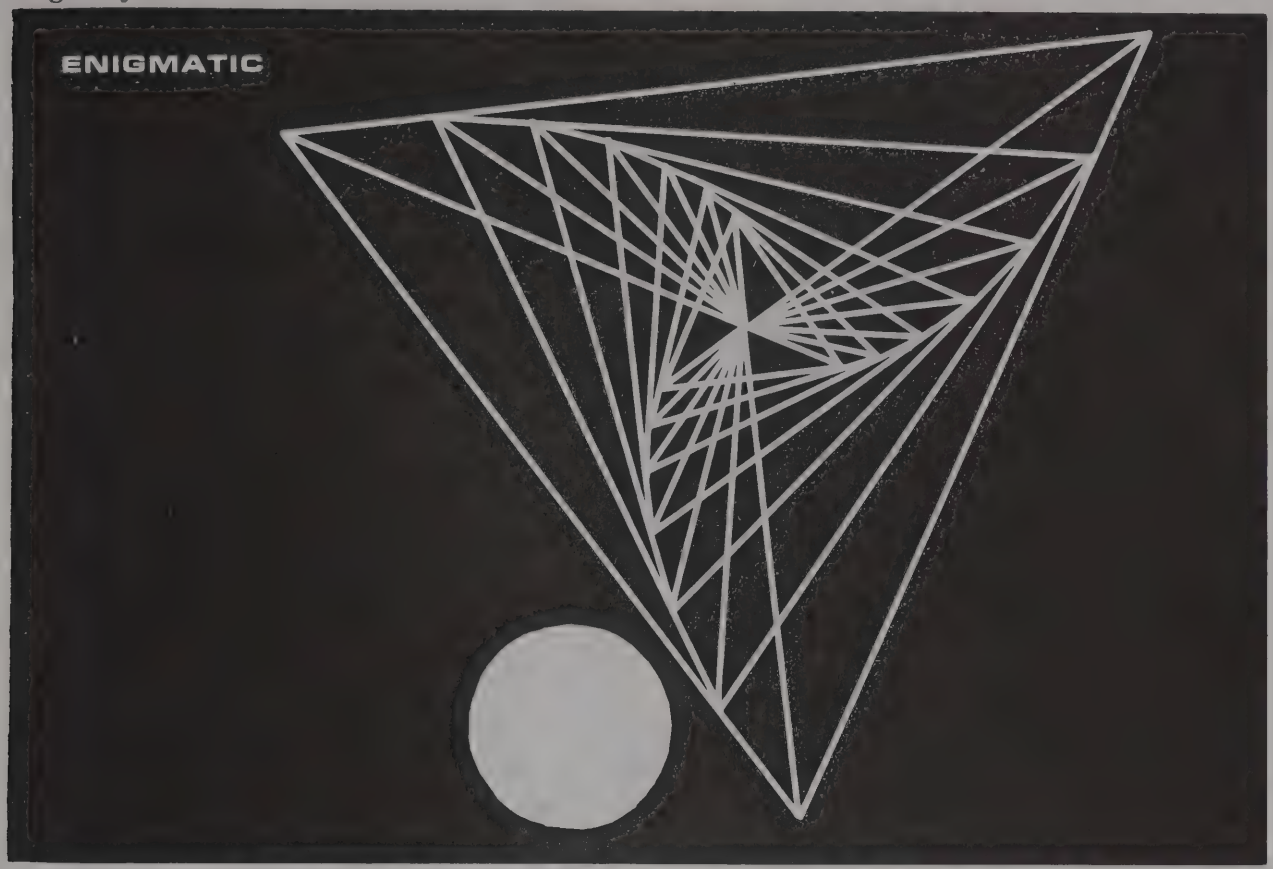
Four bugs chasing each other:



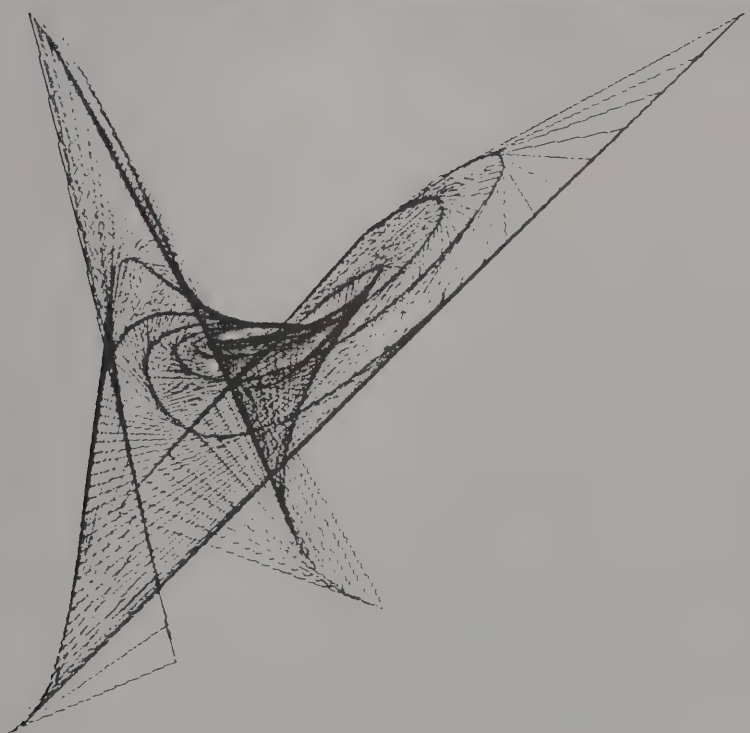
The next example, called *Compelling*, appeared in the famous book and movie, *The Dot and the Line*. (Norton Juster made it by modifying artwork that had appeared in *Scripta Mathematica*.) It resembles the example above, but makes the 4 bugs start as a rectangle (instead of a square), and makes the bug in the upper left corner chase the bug in the opposite corner (although *looking* at a nearby bug instead).



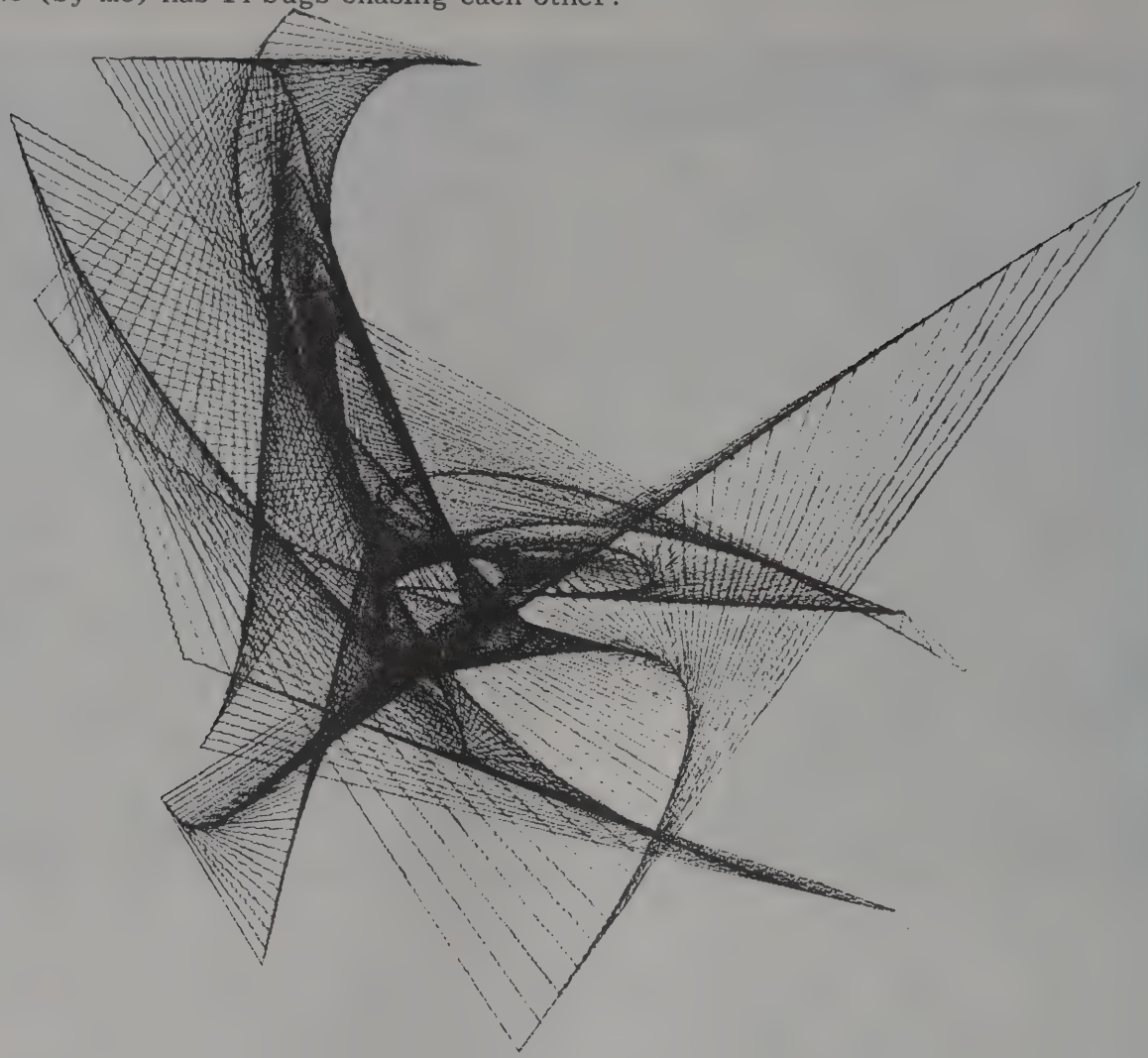
Enigmatic (from *The Dot and the Line*) makes 3 bugs chase each other, and a fourth bug stay motionless in the center:



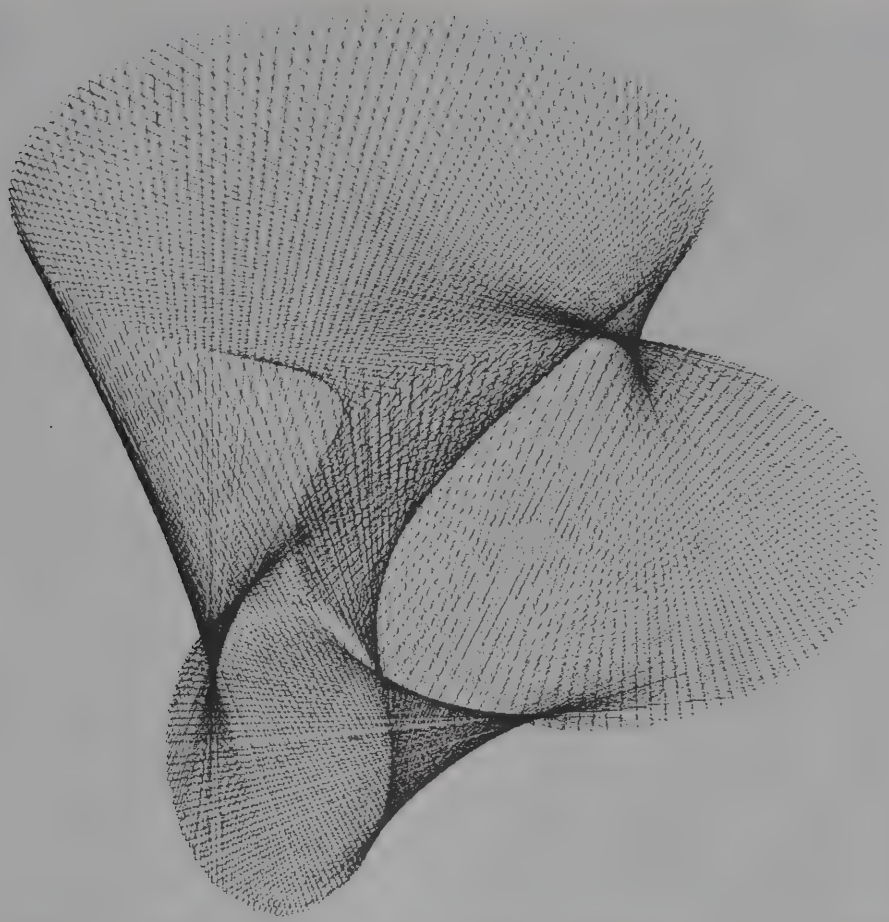
Kite (by me) has 8 bugs chasing each other:



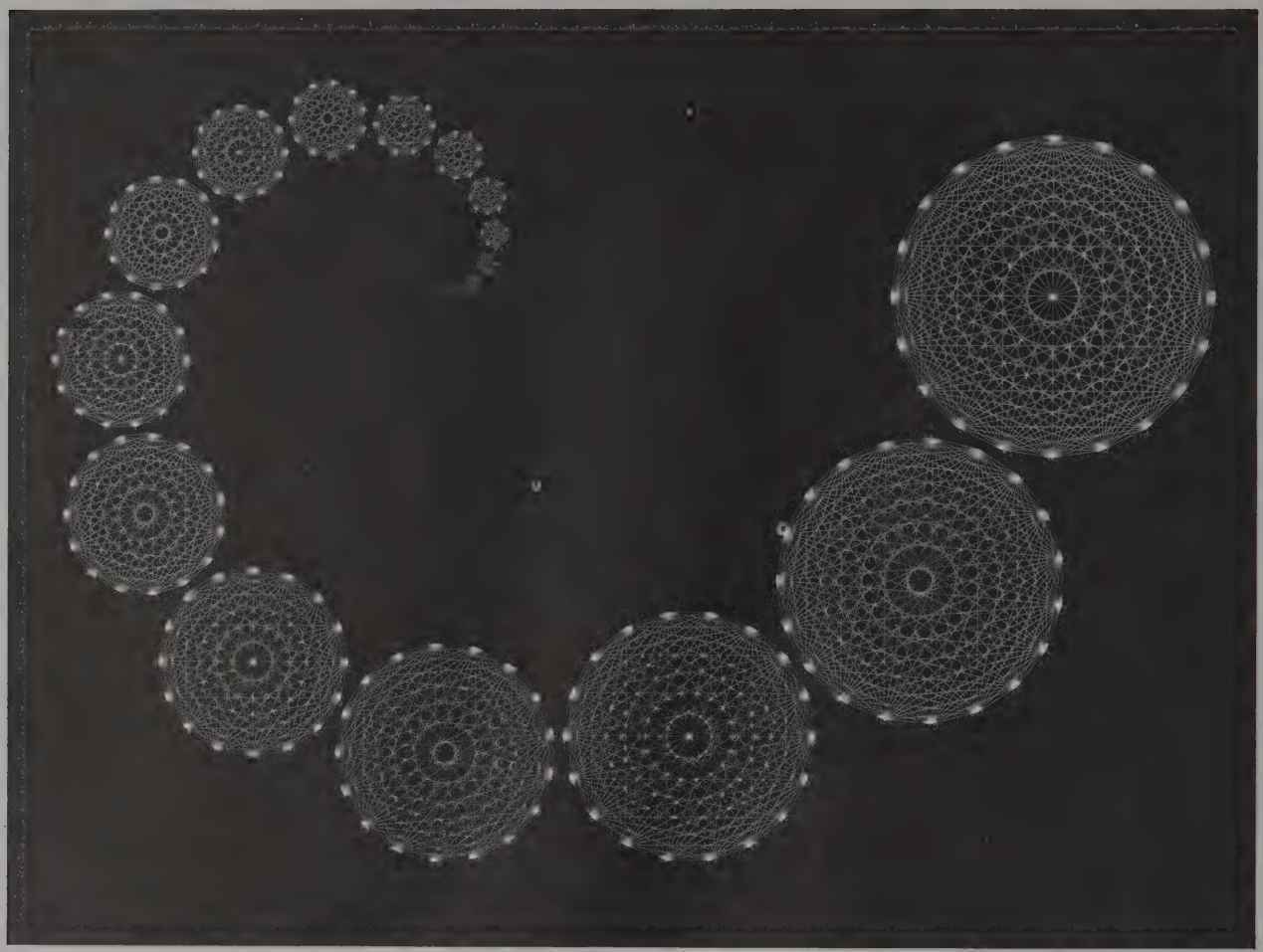
Sails (by me) has 14 bugs chasing each other:



Elliptic Motion (by my student Toby D'Oench) has 3 bugs staring at each other, as they travel on 3 elliptical highways:



Archimedean Spiral (by Norton Starr) has bugs on circles; the bugs stare at each other but don't move:



FAMOUS ART

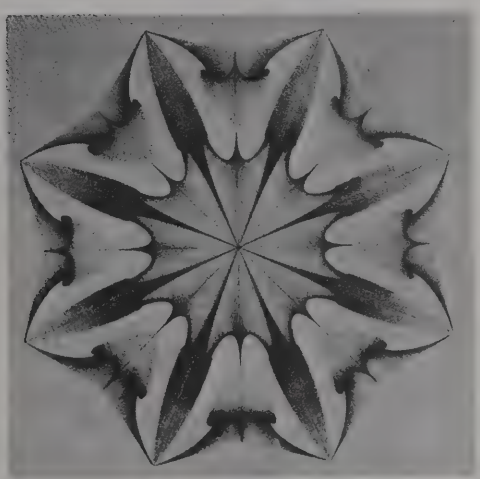
These famous works of computer art were done back in the 1960's:

Cubic Interpolation
by Schmitt & Gaines

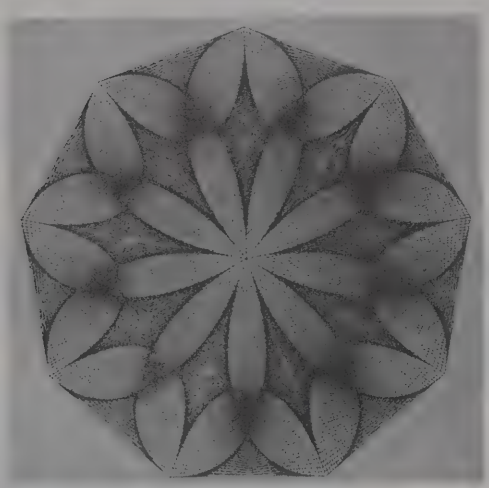
Hummingbird
by Kerry Strand

The Snail
by Kerry Strand

Krystallos
by Strand & Jenkins



Symplexity
by Strand & Jenkins



WHAT'S ART?

To create art, write a weird program whose consequences you don't fully understand, tell the computer to obey it, and look at the computer's drawing. If the drawing looks nice, keep it and call it "art"— even if the drawing wasn't what you expected. Maybe it resulted from an error, but so what? Anything interesting is art.

If the drawing "has potential" but isn't totally satisfying, change a few lines of the program and see what happens— or run the program again unchanged and hope the random numbers will fall differently. The last thing to invent is the title. Whatever the drawing reminds you of becomes the title.

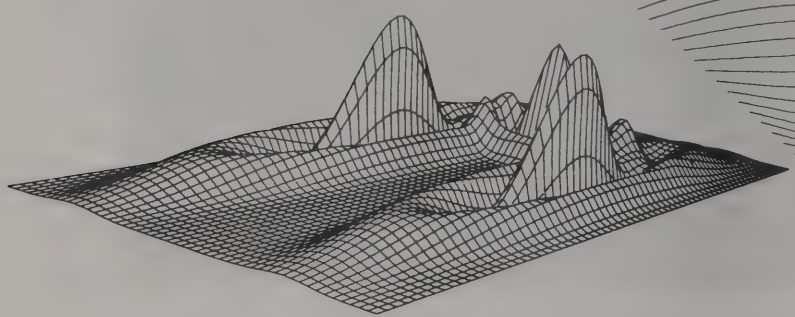
For example, that's how I produced *Kite* and *Sails*. I did *not* say to myself, "I want to draw a kite and sails". I just let the computer pick random starting points for the bugs and watched what happened. I said to myself, "Gee whiz, those drawings remind me of a kite and sails." So I named them *Kite* and *Sails*, and pretended that I chose those shapes on purpose.

That method may seem a long way from DaVinci, but it's how most computer art gets created. The rationale is: don't overplan...let the computer "do its own thing"; it will give you art that escapes from the bounds of human culture and thereby expands your horizons.

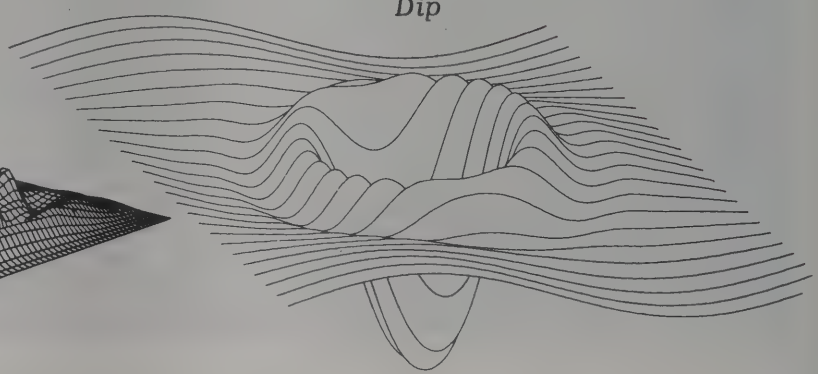
THREE-DIMENSIONAL DRAWING

The computer drew these three-dimensional surfaces:

Three Peaks
by John Szabo

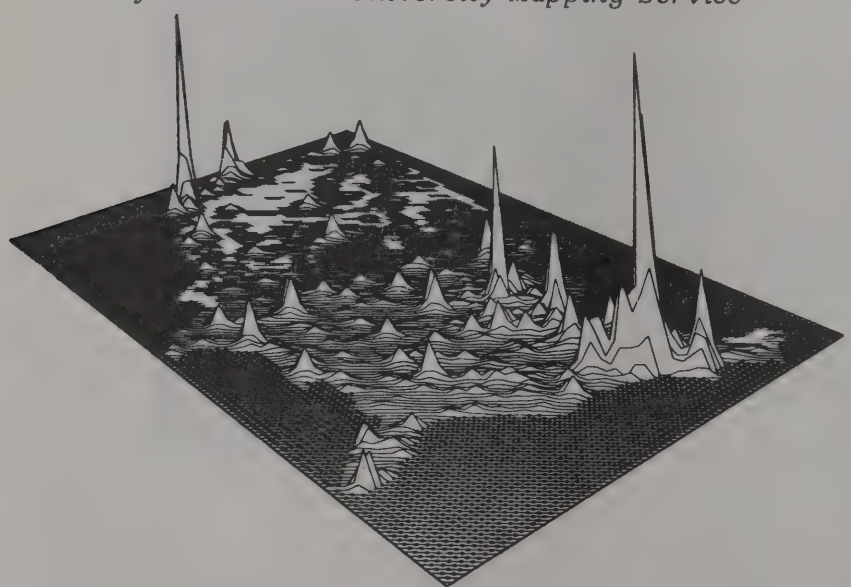


Dip

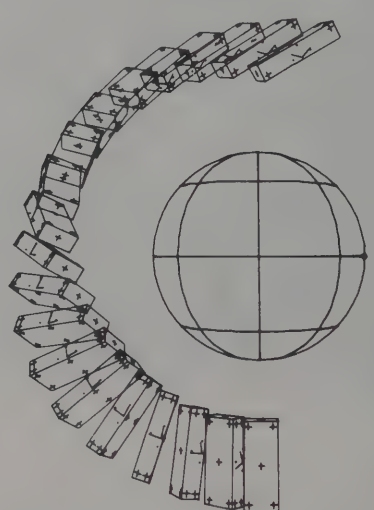


Those were done for the sake of art. These were done for the sake of science:

Population Density in the U.S.
by the Harvard University Mapping Service



Rotation and Revolution
of a Communication Satellite
by Bell Telephone Laboratories



The hardest part about three-dimensional drawing is figuring out which lines the computer should *not* show, because they're hidden behind other surfaces.

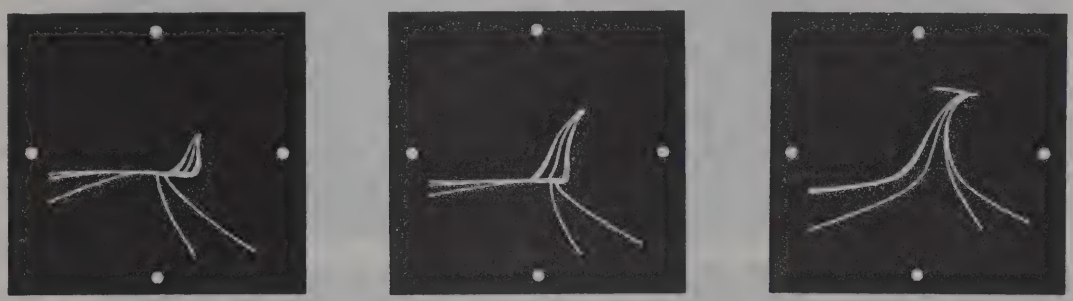
A three-dimensional drawing can become part of a much larger project--- such as building a road:

At the Drawing Board
by Nigel Hawkes (abridged)

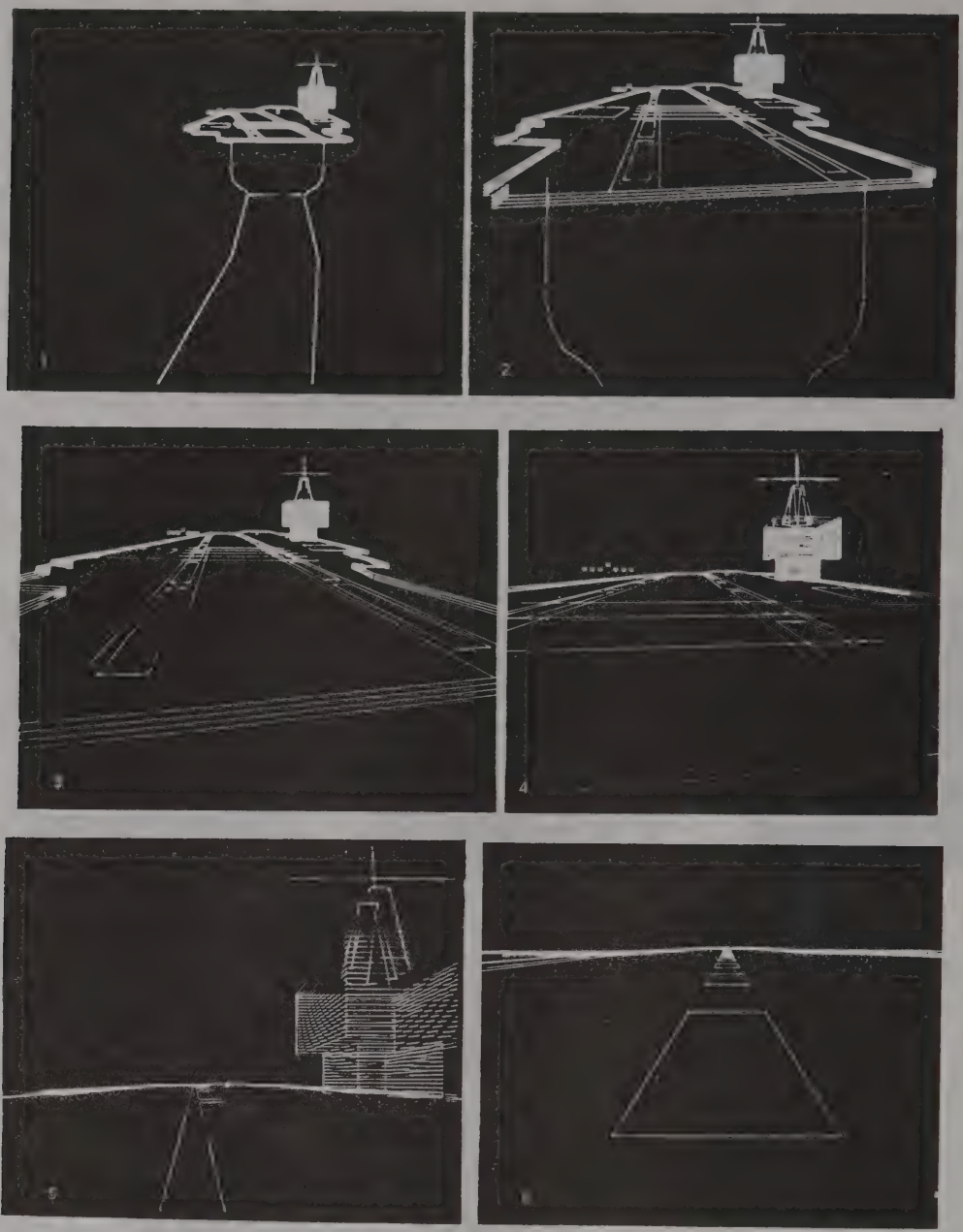
Road building is a complex operation, involving hundred of separate problems. Since 1959 the Swedish Government and the firm Nordisk ADB of Stockholm have collaborated on a program for mechanizing these calculations. As early as 1963, more than 8000 miles of road had been planned by the system, at a savings at the planning stage of 20%.

The process starts with a series of aerial photographs of the area over which the road will run. From these, the rise and fall of the land can be analyzed stereoscopically with remarkable accuracy— said to be within four inches. The information from the pictures is fed into the computer, which calculates the amount of "cut" and "fill" needed, and how earth moved from one point can be used as fill at another. The computer allows for variations in the cost of the land, in the total length of the route, and other factors. By taking these into account, it devises the cheapest possible route between two points.

On a graphics terminal, the computer draws pictures of what the road will look like at one-meter intervals. By filming each of these pictures as a separate frame, it's possible to "drive" along the imaginary road before it's built. This isn't a trivial exercise; occasionally, slopes and bends conspire to give the driver the illusion that the road is slipping sideways away from him. By observing this effect before the road's built, it can be avoided in the final design.



At the University of Utah, Ivan Sutherland made the computer show what happens when you try to land an airplane on an aircraft carrier. As your plane approaches the carrier, you can adjust its speed and angle by typing at the keyboard.



NEW STYLE

Computer art is changing. The "old style"— which you've been looking at— consists of hundreds of thin lines in mathematical patterns, drawn on paper and with little regard for color. The "new style" uses big blobs and streaks of color, flashed on a T.V. tube or film, which is then photographed.

BE POETIC

THE TAOIST HIROSHIMA ELEVATOR (1961)

In Italy, Nanni Balestrini created poetry by feeding the computer three passages:

(from Lao Tzu's *Tao Te Ching*) "While the multitude of things comes into being, I envisage their return. Although things flourish, they all return to their roots."

(from Michihito Hachiya's *Hiroshima Diary*) "The blinding fireball expands rapidly, thirty times brighter than the sun. When it reaches the stratosphere, the summit of the cloud takes on the well-known mushroom shape."

(from Paul Goldwin's *The Mystery of the Elevator*) "...head pressed on shoulder, hair between lips, lay motionless without speaking, till he moved his fingers slowly, trying to grasp."

He told the computer to mix them up. Here's the result:

Head pressed on shoulder, thirty times
brighter than the sun, I envisage their return—
until he moved his fingers slowly...And while the multitude
of things comes into being, at the summit of the cloud
they all return to their roots and take on
the well-known mushroom shape, endeavoring to grasp.

Hair between lips, they all return
to their roots. In the blinding fireball
I envisage their return— until he moved his fingers
slowly— and although things flourish
takes on the well-known mushroom shape, endeavoring
to grasp, while the multitude of things comes into being.

In the blinding fireball I envisage
their return. When it reaches the stratosphere, while the multitude
of things comes into being, head pressed
on shoulder, thirty times brighter than the sun,
they all return to their roots. Hair
between lips takes on the well-known mushroom shape.

They lay motionless without speaking. Thirty times
brighter than the sun, they all return
to their roots. Head pressed on shoulder,
they take on the well-known mushroom shape, endeavoring
to grasp. And although things flourish,
they expand rapidly, hair between lips.

While the multitude of things comes into being in the blinding
fireball, they all return
to their roots. They expand rapidly— until he moved
his fingers slowly, when it reached the stratosphere
and lay motionless without speaking, thirty times
brighter than the sun, endeavoring to grasp.

I envisage their return— until he moved his fingers
slowly in the blinding fireball.
They all return to their roots. Hair
between lips and thirty times brighter than the sun
lay motionless without speaking. They expand,
rapidly endeavoring to grasp the summit.

In Balestrini's program, *Tao Te Ching* was represented as four phrases:

while the multitude of things comes into being
I envisage their return
although things flourish
they all return to their roots

Hiroshima Diary was represented as six phrases, and *The Mystery of the Elevator* as five. For each verse, the computer was told to choose nine phrases at random, and print them in a random order (never juxtaposing phrases from the same passage), to form six lines of roughly equal metrical length. Actually the computer printed the poem in capital letters, without punctuation; Balestrini himself then added the punctuation and polished the grammar. The whole thing was actually done in Italian; you've been reading Edwin Morgan's translation, with my punctuation.

BUBBLES (1966)

At Northwestern University, programmers made the computer compose nice poetry. To use their program, you type a list of nouns, verbs, and other words. The computer randomly chooses five of your words to be *theme words*. The computer combines all your words to form sentences, but chooses the theme words more often than the others. It combines the sentences into verses and tries to keep the lengths of the lines approximately equal. It puts a theme word into the title.

In one poem, the computer chose *bubble* to be a theme word. The title was: ODE TO A BUBBLE. The poem contained phrases such as, "Ah, sweet bubble." The word *bubble* appeared so often that even the stupidest reader could say:

"Oh, yeah. I really understand this poem. Ya see, it's about a bubble."

The poem had all the familiar poetic trappings, such as "but alas!", which marked the turning point. (Cynics argue that the poem didn't *really* have a turning point, since the computer didn't have the faintest idea of what it was saying!)

KIDS AND PHYSICS (1968)

In England at Manchester University, Mendoza made the computer write children's stories. Here's a story the computer composed:

The sun shone over the woods. Across the fields softly drifted the breeze, while then the clouds, which calmly floated all afternoon, moved across the fields.

Squirrel, who scampered through the trees, quickly ran off; and off noisily ran Little Grey Rabbit. She sniffed at the house; but out of the door noisily hurried Hare, who peered at slowly the flowers. Squirrel quickly scampered over the woods and fields, but Old Grey Owl flew over the woods and fields. Down the path to the woods ran Little Grey Rabbit, who then sniffed at a strawberry pie.

The first paragraph uses these words:

	Verbs moved	drifted	shone	floated	touched	melted	looked down on	warmed
Nouns								
the clouds	1	1	0	1	0	0	1	0
the sun	0	1	1	1	1	0	1	1
the breeze	1	1	0	1	1	2	0	0
the sky	0	0	0	0	1	0	1	1
Adverbs								
gently	1	1	1	1	1	1	1	1
quietly	1	1	1	1	1	1	1	1
loudly	1	1	1	1	1	1	1	1
softly	1	1	1	1	1	1	1	1
calmly	1	1	1	1	1	1	1	1
soon	1	1	1	1	1	1	1	1
then	1	1	1	1	1	1	1	1
(no adverb)	2	2	2	2	2	2	2	2
Endings								
by	1	1	0	1				
over the woods	1	1	1	1				
across the meadows	1	1	1	1				
through the trees	1	1	1	1				
down	0	0	1	0				
for a long time	0	0	1	1				
all day	1	1	1	1				
all afternoon	1	1	1	1				
the grass					1	1	1	1
the leaves of the trees					1	1	1	1
the garden					1	1	1	1
the flowers					1	1	1	1
the little house					1	0	1	1
the old oak tree					1	1	1	1
the treetops					1	1	1	1

Additional words: which, and, while, they, it

To construct a sentence, the computer uses that table. Here's how. First, the computer randomly chooses a noun. Suppose it chooses *the sun*. Then it looks across the row marked *the sun*, to choose a verb whose score isn't 0. For example, it's possible that *the sun shone*, but not possible that *the sun melted*. Suppose it chooses *shone*. Then it looks down the column marked *shone*, to choose an adverb and an ending. Notice that the ending can't be *by*, since its score is 0. *No adverb* has a score of 2, whereas *gently* has a score of 1; that makes *no adverb* twice as likely as *gently*. If the computer chooses *no adverb* and *over the woods*, the resulting sentence is: The sun shone over the woods. In fact, that's the first sentence of the story you just read.

The computer occasionally changes the word order: instead of typing "The breeze drifted softly across the fields", the computer begins the second sentence by typing, "Across the fields softly drifted the breeze". To combine short sentences into long ones, the computer uses the words at the bottom of the table: *which*, *and*, *while*, *they*, and *it*. If two consecutive clauses have the same subject, a pronoun is substituted: *they* replaces *the clouds*; *it* replaces *the sun*, *the trees*, and *the sky*. A *which* clause can modify any noun, but can't modify a pronoun. The verb in a *which* clause must be different from the one in the main clause.

Here's the vocabulary for the second paragraph:

Nouns	Verbs: scampered	flew	ran	hurried	sniffed at	peered at	ate	munched and crunched
Little Grey Rabbit	0	0	2	3	1	1	0	0
Old Grey Owl	0	3	0	0	1	3	2	2
Squirrel	3	0	1	1	1	1	3	3
Hare	0	0	0	2	1	1	2	2
Adverbs								
then	0	1	1	1	1	1	0	0
slowly	0	2	0	0	1	1	1	1
quickly	1	1	1	1	0	0	1	1
soon	1	0	1	1	0	0	1	1
happily	1	0	0	1	0	0	1	1
gaily	1	0	0	1	0	0	1	1
noisily	1	0	1	1	0	0	2	3
(no adverb)	5	4	4	5	2	2	5	5
Endings								
off	1	1	1	1				
over the woods and fields	1	1	1	1				
through the trees	1	1	1	1				
among the treetops	0	1	0	0				
into the home	1	0	1	1				
out of the door	1	0	1	1				
down the path to the woods	1	0	1	1				
about the garden	1	1	1	1				
the house					1	1	0	0
the hollow tree					1	1	0	0
an old oak tree					1	1	0	0
the flowers					1	1	0	0
two buns					1	1	1	1
a strawberry pie					1	1	1	1
six cabbages					1	1	1	1

Additional words: who, and, but, she, he

Here's another story the program produced:

The breeze drifted by. Across the fields softly moved the clouds; and then the breeze, which calmly touched the treetops, drifted across the fields. Quietly the sun shone over the woods. The sky calmly shone across the fields.

Out of the door ran Squirrel; and off hurried Hare, who munched and crunched two buns happily. Off slowly flew Old Grey Owl, and Squirrel soon ate two buns. Old Grey Owl, who peered at a strawberry pie, munched and crunched two buns; but noisily Little Grey Rabbit, who peered at an old oak tree, slowly ran down the path to the woods. Soon she hurried down the path to the woods, but then she sniffed at two buns. She hurried down the path to the woods.

Why did Mendoza make the computer write those stories? He explains:

This work all began when a well-known scientist joined our physics department. He had spent several years away from academic life and was able to take a long cool look at academic procedures. He soon formed the theory that students never learned any ideas; all they learned was a vocabulary of okay words which they strung together in arbitrary order, relying on the fact that an examiner pressed for time would not actually read what they had written but would scan down the pages looking for these words. I set out to test his hypothesis.

I began by writing "Little Grey Rabbit" stories. I tested these stories out on my very small children; but after some minutes they grew irritable, because nothing actually happened. This shows that even small children of three can measure entropy.

Then I altered the vocabulary and grammar— making the sentences all very dead— to imitate the style of physics textbooks. The endpoint came when a colleague at another university secretly sent me an exam a week before it was given to the students. I wrote vocabularies and copied down what the computer emitted. Using a false name, I slipped my paper in among the genuine ones. Unfortunately, it was marked by a very conscientious man, who eventually stormed into the Director's office shouting, "Who the hell is this man— why did we ever admit him?" So perhaps my colleague's hypothesis was wrong, and students are a little better than we think.

Here are two of the computers answers:

In electricity, the unit of resistance is defined by electrolysis; and the unit of charge, which was fixed at the Cavendish lab in Rayleigh's classic experiments, was measured at the Cavendish lab. Theoretically, the absolute ohm is defined in a self-consistent way. The unit of resistance, which was determined with a coil spinning in a field, was fixed at the Cavendish lab; and this, by definition, is expressed in conceptual experiments. Theoretically the absolute ohm, which was redetermined using combined e.m.u. and e.s.u., is expressed by the intensity at the center of a coil.

In this country, Soddy considered Planck's hypothesis from a new angle. Einstein 50 years ago asserted quantisation. At a photocathode, electrons which undergo collisions in the Compton effect as energy packets or quanta are emitted at definite angles; nevertheless, particles in a photocell produce photoelectrons of energy $h\nu=E_0$. Photons in vacuo transmute into lower frequencies, and light quanta in the Compton effect emit emission currents. Particles emit current proportional to energy; electrons in vacuo interact with loss of surface energy (work function); nevertheless, particles which are emitted in a photocell with conservation experimentally are conserved with energy $h\nu$. The former, at a metal surface, undergo collisions with emission of current; and at a metal surface, electrons produce emission currents. Einstein assumed the gas of quantum particles; but quite recently Rayleigh, who quite recently solved the problem in an oldfashioned way, considered radition classically. Planck, who this century assumed the A and B coefficients, explained the gas of quantum particles but before Sommerfield; Rayleigh, who quite recently was puzzled on Boltzmann statistics, tackled the problem with disastrous results. Planck, who assumed the gas of quantum particles in 1905, this century considered the ultraviolet catastrophe; but quite recently Jeans, who tackled the problem in an oldfashioned way, was puzzled with disastrous results. Black body radiation that exerts thermodynamic forces in an engine is equivalent to a relativistic system. Out of a black body, a photon that is equivalent to (out of a black body) an assembly of photons is assumed to be a non-conservative system; at the same time, thermodynamically, black body radiation that in a piston is assumed to be a relativistic system exerts quantised forces. The radiation gas that obeys Wien's displacement law is considered as a system of energy levels. Quantally, a quantum particle exerts a Doppler-dependent pressure, although this produces equilibrium transition probabilities. Black body radiation in an engine produces equilibrium transition probabilities.

AEROSPACE (1968)

In 1968, Raymond Deffrey programmed the computer to write fake reports about the aerospace industry. Shortly afterwards, I improved the program. The improved program contains these lists:

Introductory phrases
in particular, on the other hand, however, similarly, as a resultant implication,
in this regard, based on integral subsystem considerations, for example, thus,
in respect to specific goals, utilizing the established hypotheses, moreover,
in addition, in view of system operation, furthermore, to some extent,
considering the postulated interrelationships, for the most part, indeed,
based in system engineering concepts

Noun phrases
a large portion of the interface coordination communication,
a constant flow of effective information, the characterization of specific criteria,
initiation of critical subsystem development, the fully integrated test program,
the product configuration baseline, any associated supporting element,
the incorporation of additional mission constraints, the independent function principle,
a consideration of system and/or subsystem technologies, the sophisticated hardware,
the anticipated third-generation equipment, the subsystem compatibility testing,
the structural design, the preliminary qualification limit,
the philosophy of commonality and standardization,
the evolution of specifications over a given time, the greater flight-worthiness concept,
any discrete configuration made, the total system rationale

Verb phrases
must utilize and be functionally interwoven with,
maximizes the probability of success and minimizes time for,
adds explicit performance limits to, necessitates that urgent consideration be applied to,
requires considerable systems analysis to arrive at,
"is further compounded, when taking into account",
presents extremely interesting challenges to,
recognizes the importance of other systems and necessity for,
effects a significant implementation to, adds overriding performance constraints to

To produce a typical sentence, the computer prints an introductory phrase, then a noun phrase, then a verb phrase, then a noun phrase. The phrases are chosen randomly.

Each paragraph consists of six such sentences. The computer isn't allowed to use the same phrase twice within a paragraph. The introductory phrase is omitted from the first sentence of the first paragraph, the second sentence of the second paragraph, etc.; so the report can't begin with the word *furthermore*, and the style varies.

Here's the beginning of one such report:

The Economic Considerations of the Aerospace Industry

A large portion of the interface coordination communication necessitates that urgent consideration be applied to the product configuration baseline. For example, the fully integrated test program adds explicit performance limits to the independent function principle. Moreover, the sophisticated hardware presents extremely interesting challenges to the philosophy of commonality and standardization. In view of system operation, a constant flow of effective information must utilize and be functionally interwoven with the preliminary qualification limit. In addition, any discrete configuration made adds overriding performance constraints to any associated supporting element. Thus, the anticipated third-generation equipment maximizes the probability of success and minimizes time for the total system rationale.

Moreover, the total system rationale requires considerable systems analysis to arrive at the characterization of specific criteria. Any associated supporting element necessitates that urgent consideration be applied to the structural design. In this regard, the product configuration baseline is further compounded, when taking into account initiation of critical subsystem development. On the other hand, the preliminary qualification limit adds overriding performance constraints to the greater flight-worthiness concept. In respect to specific goals, the fully integrated test program effects a significant implementation to the philosophy of commonality and standardization. In view of system operation, the independent function principle recognizes the importance of other systems and necessity for the anticipated third-generation equipment....

ME-BOOKS (1972)

In 1972, Freeman Gosden Jr. started the Me-Books Publishing Company. It published books for kids. But if you bought a Me-Book for your child, you wouldn't see in it the traditional names "Dick, Jane, and Sally"; instead, you'd see the name of your own child. To order the book, you had to tell the company the names of all your children, and their friends, and pets. Their names appeared in the story.

The story was printed beautifully, in a 32-page hard-covered book with pictures in color. It cost just $3.95.

You could choose from four stories: "My Friendly Giraffe", "My Jungle Holiday", "My Birthday Land Adventure", and "My Special Christmas".

For example, if you lived on Jottings Drive, and your daughter's name was Shea, and her friend's name was Douglas, the story "My Friendly Giraffe" included paragraphs such as this:

One morning Shea was playing with Douglas in front of her home. When she looked up, what do you think she saw walking down the middle of Jottings Drive? You guessed it. A giraffe!

Ted Nelson, author of *Computer Lib*, played a trick. He ordered a copy of "My Friendly Giraffe", but pretended that his child's name was "Tricky Dick Nixon" who lived on "Pennsylvania Ave." in "Washington". Sure enough, the company sent him "My Friendly Giraffe: A Me-Book for Tricky Dick". Here are some excerpts:

Once upon a time, in a place called Washington, there lived a little boy named Tricky Dick Nixon. Now, Tricky Dick wasn't just an ordinary little boy. He had adventures that other little boys and girls just dream of. This is the story of one of his adventures. It's the story of the day that Tricky Dick met a giraffe....

As the giraffe came closer and closer, Tricky Dick started to wonder how in the world he was going to look him in the eye....

Tricky Dick knew there were no jungles in Washington. Especially on Pennsylvania Ave. But Tricky Dick wasn't even a little bit worried. First, because he was a very brave little boy. And second, because he knew that his friend, the giraffe, would never take him anyplace bad....

Tricky Dick was home. Back in Washington. Back on Pennsylvania Ave. And with a story to tell his friends, that they wouldn't have believed if they hadn't seen Tricky Dick riding off on the giraffe's back. Tricky Dick would long be a hero to this who had seen him that day....

There would be many other exciting adventures for Tricky Dick and his friends. And And maybe, just maybe, if you're a very good boy, someday we'll tell you about those,

Me-Books were commercially successful: many thousands of them were sold. Teachers liked the books, because the books encouraged students to read: students enjoyed reading books that contained their own names.

PORNOGRAPHY (1973)

I programmed the computer to turn words into sentences. For example, Judith Ann Abrams fed the computer this list of words:

Adjectives
bitchy, creepy, dead, enthusiastic, evil, gaudy, green, gross, horny, insane, juicy, mixed-up, monstrous, multicolored, neurotic, obscene, paranoid, prickly, profane, provocative, repulsive, sensuous, sexual, sexy, shiny, sick, stimulating, strange, succulent, tasty, unsettled, weird

Names
Barbara, Buster, Clyde, Colin, Debbie, Ed Shanahan, Jeff, Keith, Kerri, Liz, Miles, Nicholson 700, Norwalk

Nouns that aren't names
bastard, bikini, bitch, boat, cigarette, creep, dress, ethanol, eyelid, fly, gambling, jerk, knee, kneecap, lip, mouth, navel, piece, pornography, shirt, stud, tongue, virgin, whiskey sour, whore

Verbs that take an object
bite, caress, castrate, climb, enjoy, kiss, knife, lick, pick, pull, rip, screw, shake, take off, teach, undo

Verbs that don't take an object
disapprove, fall down, fly, hesitate, sing, sleep, sleepwalk, sneeze, vanish, walk

Exclamations
rats, such language, ugh

Here's what the computer printed:

A Sexy, Profane Creep's Tongue
by Judith Ann Abrams and the computer

The lip is tasty.

Ugh!

The mouth is enthusiastic. Does the gaudy bastard sleepwalk?

The tongue's whore caresses the bikini. When Nicholson 700 is screwing the creepy jerk over a piece, Jeff is teaching the gambling. Does paranoid Liz hesitate? Why does the tongue kiss the creep stimulatingly on horny Buster?

Undo! Pull the fly repulsively! A multicolored, enthusiastic cigarette is neurotic. Is a multicolored, weird dress's navel under the strange bikini? Why is the obscene dress's cigarette Colin's ethanol? Rats! Why is the dead, insane, tasty, shiny, knifing creep's bitch unsettled and strange?

Clyde is over Ed Shanahan. Kissing the bastard, a paranoid ethanol is unsettled. Why is the prickly kneecap's mouth on the creepy piece? Teaching Norwalk is sneezing paranoidly and grossly under provocative, shiny Debbie. As a sexy, walking tongue's virgin takes off the bitchy cigarette over the profane stud then castrates enthusiastic Miles's lip paranoidly, the dead, gaudy knee is stimulating and multicolored. Even though the dead, insane piece licks strangely and neuroticly in the flying, mixed-up eyelid's knee, the dead, biting, obscene bikini is on a repulsive mouth's gambling.

The pornography is gaudy. Kerri sleepwalks. Why is the tongue sensuous? Buster is sick. Is Miles monstrous? Debbie is neurotic and paranoid, when a stimulating fly picks the navel's jerk under Ed Shanahan. Why is the dress succulent? Hesitating, a kneecap sleeps and climbs the dead, bitchy ethanol. As insane Colin's bastard falls down weirdly in a sensuous dress, green, unsettled Miles's virgin is strange and sexual.

Is the creepy eyelid provocative? The gambling's whisky sour teaches a navel.

Is the gambling evil? The bitch walks. Is the virgin profane? Why is the navel sick? Is Liz enthusiastic? Debbie enjoys the creep. Fly! Shaking, green Kerri pulls weird Colin's fly on a sick navel, then vanishes over Norwalk.

Undo a virgin! While obscene Liz is juicy and sexual, profane, gaudy Jeff's knee is under a succulent whore's navel. Tear Keith's lip bitchily and juicily on sick, weird, multicolored Barbara! Why is Buster insane? The shirt knifes the bikini. Colin shakes the bitch. The whiskey sour hesitates over the green jerk. When a tasty tongue's ethanol walks, Kerri rips the boat and disapproves under enthusiastic Miles. Such language! Keith sings. Why is Buster bitchy?

Notice that the computer turned her adjectives into adverbs, by adding *ly* and making other changes. *Gross* became *grossly*, and *juicy* became *juicily*. Unfortunately, the computer's method wasn't perfect: the computer turned *stimulating* into *stimulatingly* (a non-existent word), and turned *neurotic* into *neuroticly* (instead of *neurotically*).

It conjugated her verbs. *Screw* became *screwing*, and *bite* became *biting* (the computer dropped the *e*). *Lick* became *licks*, and *teach* became *teaches*(the computer added the *e* after the *ch*).

It added *'s* to her nouns. *Jeff* became *Jeff's*. *Miles* became *Miles's* (it should have become *Miles'*).

For each sentence, the grammar is chosen randomly. The chance is 10% that the sentence will begin with an exclamation. If the sentence isn't merely an exclamation, the chance is 18% that the sentence will be a question.

If it's a question, there's a 40% chance it will begin with the word *why*. There's a 50% chance the main part of the question will have the form *does*...noun phrase...verb phrase, and a 50% chance it will have this form instead: *is*...noun phrase...complement.

To construct a noun phrase from nouns, adjectives, etc., the computer uses random numbers. It uses random numbers to also construct verb phrases and complements.

The program uses a special variable, called W. At the beginning of the composition, W is near zero; but it tends to increase as the composition progresses. It affects the *complexity*. When W is large, the chance of printing adjectives, adverbs, subordinate adverbs, subordinate clauses, and correlative clauses. This sentence was produced by a small W: *The lip is tasty*. This sentence was produced by a large W: *As a sexy, walking tongue's virgin takes off the bitchy cigarette over the profane stud then castrates enthusiastic Miles's lip paranoidly, the dead, gaudy knee is stimulating and multicolored.*

POETIC IMAGES (1973)

One of my students, Toby D'Oench, made the computer create poetic images, such as these:

TO GUINEVERE— LADY OF THE LAKE

Silent mists
Billow in creations
Windmills for flames evolve into ethers
Merlin again

MY MEMORY

Frozen children
Quiver with leaves
Creations with leaves hover over thoughts
Gardens of verse

A NEW ENGLAND BARN

Lazy fragrances
Waft by ethers
Seas on fragrances billow in sorrow
Rusted pitchforks

NEWPORT

Frozen sails
Slumber in fog
Hazes for sails waft by thoughts
Docks— yachts— luxuries of eras gone by

The program contains these lists:

Adjectives
fleeting, crimson, silent, sensate, pliant, gloomy, pallid, inky, frozen, lazy

Prepositions
of, on, under, above, below, in, with, by, for, through

Verbs
billow in, glitter with, flutter by, drift with, flow into, ponder about, waft by, quiver with, hover over, gleam like, wander through, slumber in, dart by, evolve into, sing to

Title...noun...ending
TO REMBRANDT...windmills...A simple brush
WAITING FOR THE PATIENT...ethers...Waiting
THE PROPHET...visions...Then a word
LISTERINE...breaths...Plastic society
EINSTEIN...thoughts...Somehow through this— and understanding of a superior order
NEWPORT...sails...Docks— yachts— luxuries of eras gone by
EXISTENCE...seas...In the beginning?
SUMMER IN WATTS...flames...Tar-street neon— and the night
TO GUINEVERE— LADY OF THE LAKE...mists...Merlin again
NOON IN CALCUTTA...hazes...Emaciated dark forms strewn like garbage
WEST HARBOR...fog...A solitary gull slices through
A NEW ENGLAND BARN...fragrances...Rusted pitchforks
A CHILD'S MICROSCOPE...creations...The wonderful amoeba
A GROUP PORTRAIT...bundles...Christmas
THE MILKY WAY...cosmos...A gooey mess
TOMBSTONE...sorrow...Rubbings
LIFE AT THE END OF A BRANCH...leaves...Swirling to the ground
SEASHELLS AND THINGS...waves...Dribble-dribble-dribble castle
A BEAVER POND...reeds...Thwack
MY MEMORY...children...Gardens of verse

To create a poetic image, the computer fills in this form:

TITLE

Adjective Noun that goes with the title
Verb Noun
Noun Preposition Noun Verb Noun
Ending that goes with the title

CURSES (1978)

Tom Dwyer & Margot Critchfield made the computer curse you. Here are some of the computer's curses:

May an enraged camel overwhelm your garage.
May an ancient philosopher lay an egg on your dill pickle.
May seven large chickens sing an operatic solo to your love letters.

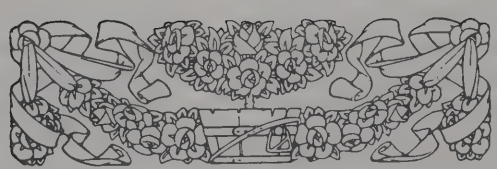
To invent a curse, the computer fills in the blanks:

May _____ _____ your _____.
subject verb phrase object

The computer uses these words randomly:

Subjects	Verb phrases	Objects
an enraged camel	send a mash note to	mother-in-law
an ancient philosopher	get inspiration from	psychoanalyst
a cocker spaniel	redecorate	rumpus room
the Eiffel Tower	become an obsession of	fern
a cowardly moose	make a salt lick out of	garage
the silent majority	buy an interest in	love letters
the last picture show	overwhelm	piggy bank
a furious trumpet player	pour yogurt on	hamburger
Miss America	sing an operatic solo to	dill pickle
seven large chickens	lay an egg on	Honda

You can find that program on page 152 of their book, *BASIC and the Personal Computer*.



ANALYZE WRITING

ENGLISH POETRY

From 1957 to 1959 at Cornell University, Stephen Parrish made the computer print an alphabetical list of the words in Matthew Arnold's poetry. Here's an excerpt:

	Page in book	Poem's title	Line in poem
CONSCIOUS			
back with the conscious thrill of shame	181	Isolation Marg	19
conscious or not of the past	287	Rugby Chapel	45
CONSCIOUSNESS			
the last spark of man's consciousness with words	429	Empedocles II	30
and keep us prisoners of our consciousness	439	Empedocles II	352
CONSECRATE			
Peter his friend with light did consecrate	445	Westmin Abbey	50
CONSECRATES			
which consecrates the ties of blood for these indeed	196	Frag Antigone	31
CONSECRATION			
won consecration from time	281	Haworth Church	46
foreshown thee in thy consecration-hour	446	Westmin Abbey	75
CONSIDER			
consider him consider not old hates	335	Merope	173
consider him consider not old hates	335	Merope	173
consider too this people who were dear	335	Merope	174

To find out what Matthew Arnold said about love, just look up LOVE. Such an index is called a *concordance*.

Joseph Raben (at Queens College) believe the poet Shelley borrowed imagery from Milton. In 1964, he made the computer produce a concordance of Shelley's *Prometheus Unbound* and a concordance of Milton's *Paradise Lost* and compare them. The computer found many similarities between Shelley and Milton.

In 1971 at Münster University in Germany, Marvin Spevack fed the computer all the works of Shakespeare, and made it count how often each word occurs. Disregarding trivial words such as *a* and *the*, the computer discovered Shakespeare's favorite word was *love*: he used it 2,271 times. Next come *heart*, *death*, *man*, *life*, and *hand*. He never used the word *hero*. In *Macbeth*, the word *good* occurs more often than any other adjective, noun, or adverb, and more often than most verbs. Using similar techniques, other researchers have made the computer graph the rise and fall of themes in a novel.

AMERICAN HISTORY

In 1964, Mosteller and Wallace made the computer discover who wrote the Federalist papers. Historians had known some of the papers were by Alexander Hamilton and others by James Madison; but the authorship of the remaining papers had been in dispute. The computer compared the literary styles of the papers, by counting the frequency of words such as *by*, *enough*, *from*, *to*, *upon*, *while*, and *whilst*. It concluded that all the disputed papers were written by Madison, not Hamilton. The statistical evidence was so high that historians accept the computer's finding as fact.

THE BIBLE

The most important computer analysis of the Bible was done by John Ellison, a Texas clergyman. In 1951, he made the computer compare 309 Greek manuscripts of the New Testament. Underneath each word of a standard text, the computer printed the variants found in the other manuscripts. It classified the manuscripts according to their similarities.

In 1957, he published a concordance to the Revised Standard Bible, and a pair of other researchers (Tasman & Busa) indexed the Dead Sea Scrolls.

In 1964, Andrew Q. Morton (a Scottish clergyman) used the computer to deduce that Paul didn't write some of the Biblical letters attributed to him. All Morton did was count the number of times Paul used the Greek word *kai* in each sentence. *Kai* means *and*. Coming to a conclusion about Biblical authorship by counting just the word *and* may seem silly; but Morton said he analyzed 20 writers of ancient Greek and found each used *kai* with a constant frequency; in the "Pauline" letters, the frequency of *kai* varied considerably, implying some of them were not by Paul.

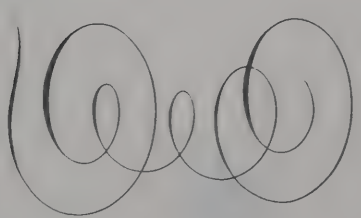
Ellison distrusted Morton's assumption that a man's literary style must remain constant. He warned: if Morton's method were applied to the Declaration of Independence and Jefferson's letters to his wife, the computer might conclude that either Jefferson didn't write the Declaration of Independence or another man was writing love letters to Mrs. Jefferson. In 1965, to prove his point, he applied Morton's method to two of Morton's own articles on the subject: the computer concluded that Morton could not be the author of both!

FORGERY

Recently, IBM programmed one of its computers to detect a forged signature— even if the signature looks correct to the naked eye. To use the IBM forgery-detection system, write your signature by using a special IBM pen, attached to the computer. As you write, the computer keeps noticing how hard you press the pen against the paper, and how fast you accelerate the pen's motion.

If somebody else tries to pretend he's you, he must sit down at the machine and try to duplicate your signature. If he presses the pen hardest at different points of the signature, or if he accelerates at different points, the computer says he's a fake. The system works well, because the average crook trying to forge your signature will hesitate at the hard parts; his hesitations affect the pen's pressure and acceleration, which tell the computer he's faking.

IBM isn't selling the system to the public yet. And when it does, remember: the system works only on signatures written with IBM's pen.



TRANSLATE RUSSIAN

WHY RUSSIAN?

Soon after computers were invented, programmers tried to make them translate Russian into Englsh. I suspect Russian was chosen instead of French for three reasons:

1. Few humans could translate Russian. French translators were a-dime-a-dozen.
2. Computer experts love hard problems. Russian is harder than French.
3. Most computers were owned by the Department of Defense, which is *very* interested in Russia.

EARLY ATTEMPTS

In 1954, IBM wrote a program that translated Russian sentences such as:

Gasoline is prepared by chemical methods from crude oil. The price of crude oil is determined by the market. The quality of the crude oil is determined by the calorie content.

Unfortunately, most Russian sentences are not so simple. During the 1960's, the end of a Russian paper on space biology was fed into an advanced program written by Computer Concepts, Inc. Here's the translation that came out:

Thus, the examination of some from fundamental RADIOBIOLOGICESKIX problems shows, that in this a field still very much NEREWENNYX questions. This is clear, since cosmic RADIOBIOLOGI4 is very young RAZDELOM young science efforts of scientific different specialties of the different countries of the world successful PRODOLJENY will be expanded there are.

The computer couldn't translate the words in capital letters and was stumped by Russian grammar. A competing program, written by the Air Force, translated the same passage a little better:

Thus, consideration of from basic radio-biological problems shows that in a given region still very many unsolved questions. This and intelligibly, since space radiobiology is very young division of young science— space biology. However is base to trust that jointly scientists of different specialties of various countries of world/peace radiobiological investigations in outer space will be successfully continued and expanded.

In 1966, a special committee of the National Academy of Sciences concluded that the experience of computer translation was "uniformly discouraging" and that it was cheaper to hire a human translator than to do the two-step process of computer translation followed by human editing.

FAMOUS ERRORS

If you program the computer to translate an English sentence into Russian, and then the Russian back to English, will you get back the same English sentence you started with?

One programmer tried, "The spirit is willing, but the flesh is weak." The computer translated it into Russian, then back into English, and printed, "The booze is strong, but the meat is rotten."

Another programmer tried, "Out of sight, out of mind." The computer printed, "Blind idiot."

At an engineering conference, a computer was translating scientific papers into English, when it suddenly started talking about "water sheep". Everyone was confused. Finally they figured it out— the computer meant *hydraulic rams*.

LOOK BACK

DREAMS

For many centuries, people have dreamed of inventing a computerized machine that acts like a person. Let's look at those dreams. . . .

THE GREEKS

The hope of making an inanimate object act like a person can be traced back to the ancient Greeks. According to Greek mythology, Pygmalion sculpted a statue of a woman, fell in love with it, and prayed to the gods to make it come to life. His wish was granted—she came to life. And they lived happily ever after.

RAMON LULL (1272 A.D.)

In 1272 A.D. on the Spanish island of Majorca, Ramon Lull invented the idea of a machine that would produce *all* knowledge, by putting together words at random. He even tried to build it.

Needless to say, he was a bit of a nut. Here's a description of his personality (written by Jerry Rosenberg, abridged):

Ramon Lull married young and fathered two children—which didn't stop him from his courtier's adventures. He had an especially strong passion for married women. One day as he was riding his horse down the center of town, he saw a familiar woman entering church for a High Mass. Undisturbed by this circumstance, he galloped his horse into the cathedral and was quickly thrown out by the congregants. The lady was so disturbed by his scene that she prepared a plan to end Lull's pursuit once and for all. She invited him to her boudoir, displayed the bosom that he had been praising in poems written for her, and showed him a cancerous breast. "See, Ramon," she said, "the foulness of this body that has won thy affection! How much better hadst thou done to have set thy love on Jesus Christ, of Whom thou mayest have a prize that is eternal!"

In shame Lull withdrew from court life. On four different occasions a vision of Christ hanging on the Cross came to him, and in penitence Lull became a dedicated Christian. His conversion was followed by a pathetic impulse to try to convert the entire Moslem world to Christianity. This obsession dominated the remainder of his life. His "Book of Contemplation" was divided into five books in honor of the five wounds of Christ. It contained forty subdivisions—for the forty days that Christ spent in the wilderness; 366 chapters—one to be read each day and the last chapter to be read only in a leap year. Each of the chapters had ten paragraphs to commemorate the ten commandments; each paragraph had three parts to signify the trinity—for a total of thirty parts a chapter, signifying the thirty pieces of silver.

In the final chapter of his book he tried to prove to infidels that Christianity was the only true faith.

Several centuries later—in 1726—Lull's machine was pooh-poohed by Jonathan Swift, in *Gulliver's Travels*. Gulliver meets a professor who has built such a machine. The professor claims his machine lets "the most ignorant person. . . write books in philosophy, poetry, politics, law, mathematics, and theology without the least assistance from genius and study." The machine is huge—20 feet on each side—and contains all the words of the language, in all their declensions, written on scraps of paper that are glued onto bits of wood connected by wires. Each of the professor's 40 students operates one of the machine's 40 cranks. At a given signal, every student turns his crank a random distance, to push the words into new positions. Gulliver reports: "he then commanded six and thirty of the lads to read the several lines softly as they appeared upon the frame; and where they found three or four words together that might make part of a sentence, they dictated to the four remaining boys who were scribes. . . . Six hours a day the young students were employed in this labor, and the professor showed me several volumes in large folio already collected, of broken sentences, which he intended to piece together, and out of those rich materials to give the world a complete body of all arts and sciences. . . ."

KAREL CAPEK (1920)

The word *robot* was invented in 1920 by Karel Capek, a Czech playwright. His play "R.U.R." shows a factory where the workers look human but are really machines. The workers are dubbed *robots*, because the Czech word for *slave* is *robotnik*.

His play is pessimistic. The invention of robots causes unemployment. Men lose all ambition—even the ambition to raise children. The robots are used in war, go mad, revolt against mankind and destroy it. In the end only two robots are left. It's up to them to repopulate the world.

ISAAC ASIMOV (1942)

Many sci-fi writers copied Capek's idea of robots, with even more pessimism. An exception was Isaac Asimov, who depicted robots as being loving. He coined the word *robotics*, which means the study of robots, and in 1942 developed that he calls the "Three Laws of Robotics". Here's the version he published in 1950:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence, as long as such protection does not conflict with either the First or the Second Law.

NORBERT WIENER (1947)

The word *cybernetics* was invented in 1947 by Norbert Wiener, an MIT professor. He defined it to be "the science of control and communication in the animal and the machine." Wiener and his disciples, who called themselves *cyberneticists*, wondered whether it would be possible to make an electrical imitation of the human nervous system. It would be a "thinking machine". They created the concept of *feedback*: animals and machines both need to perceive the consequences of their actions, to learn how to improve themselves. For example, a machine that is producing parts in a factory should examine the parts it has produced, the heat it has generated, and other factors, to adjust itself accordingly.

Wiener, like Ramon Lull, was somewhat strange. He graduated from Tufts College when he was 14 years old, got his doctorate from Harvard when he was 18, and became the typical "absent-minded professor". Many anecdotes are told about him.

For example, once he went to a conference and parked his car in the parking lot. When the conference was over, he went to the lot, but forgot where he parked his car, and even forgot what it looked like. So he waited until all the other cars were driven away, and took the car that was left.

When he and his family moved to a new house a few blocks away, his wife gave him written directions on how to reach it, because she knew he was very absent-minded. But sure enough, when he was leaving his office at the end of the day, he couldn't remember where he put her note, and of course he couldn't remember where the new house was. So he drove to his old neighborhood instead. He saw a young child and asked her, "Little girl, can you tell me where the Wieners have moved to?" "Yes, Daddy," came the reply, "Mommy said you'd probably be here, so she sent me over to show you the way home."

One day he was sitting in the campus lounge, intensely studying a paper on the table. Every now and then, he would get up, pace a bit, and then return to the paper. Everyone was impressed by the enormous mental effort reflected on his face. Once again he rose from his paper, took a few rapid steps around the room, and collided with a student. The student said, "Good afternoon, Professor Wiener." Weiner stopped, stared, clapped a hand to his forehead, said "Wiener—that's the word," and ran back to the table to fill the word "wiener" in the crossword puzzle he was working.

Once he drove 150 miles to a math conference at Yale University; but when the conference was over, he forgot he had come by car, so he returned by bus. The next morning, he went out to his garage to get his car, discovered it was missing, and complained to the police that while he was away someone had stolen it.

Those anecdotes were collected by Howard Eves, a math historian.

ALAN TURING (1950)

Can a computer "think"? In 1950, Alan Turing proposed the following test. In one room, put a human and a computer. In another room, put another human (called the Interrogator) and give him two terminals—one for communication with the computer, and the other for communication with the other human—but don't tell the Interrogator which terminal is which. If he can't tell the difference, the computer's doing a good job of imitating the human, and, according to Turing, we should say that the computer can "think".

It's called the Imitation Game. The Interrogator asks questions. The human witness answers honestly. The computer pretends to be human.

To win, the computer must be able to imitate human weaknesses as well as strengths. For example, when asked to add two numbers, it should pause before answering, as a human would. When asked to write a sonnet, a good imitation-human answer would be, "Count me out on this one. I never could write poetry." When asked "Are you human", the computer should say "yes".

Such responses wouldn't be hard to program. But a clever Interrogator could give the computer a rough time, by requiring it to analyze its own thinking:

INTERROGATOR: In the first line of your sonnet which reads "Shall I compare thee to a summer's day," wouldn't "a spring day" do as well or better?
WITNESS: It wouldn't scan.
INTERROGATOR: How about "a winter's day". That would scan all right.
WITNESS: Yes, but nobody wants to be compared to a winter's day.
INTERROGATOR: Would you say Mr. Pickwick reminded you of Christmas?
WITNESS: In a way.
INTERROGATOR: Yet Christmas is a winter's day, and I don't think Mr. Pickwick would mind the comparison.
WITNESS: I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

If the computer could answer questions that well, the Interrogator would have a hard time telling it wasn't human.

Donald Fink has suggested that the Interrogator say, "Suggest an unsolved problem and some methods for working toward its solution," and "What methods would most likely prove fruitful in solving the following problem. . . ."

Turing believed computers would someday be able to win the game and therefore be considered to "think". In his article, he listed possible objections to his belief, and rebutted them. . . .

OBJECTION: Thinking's a function of man's immortal soul. Since computers don't have souls, computers can't think.
REBUTTAL: Since God's all-powerful, He can give computers souls if He wishes. Just as we create children to house His souls, so should we serve Him by creating computers.

OBJECTION: If machines could equal us in thinking, that would be dreadful!
REBUTTAL: Too bad!

OBJECTION: Logicians have proved it's impossible to build a computer that can answer every question.
REBUTTAL: Is it possible to find a *human* that can answer every question? Computers are no dumber than us. And though no one can answer every question, why not build a succession of computers, each one more powerful than the next, so every question could be answered by at least one of them?

OBJECTION: Although computers can produce, they can't be *conscious* of what they've produced. They can't feel pleasure at their successes, misery at their mistakes, and depression when they don't get what they want.
REBUTTAL: The only way to be absolutely sure whether a computer has feelings is to become one. A more practical experiment would be to build a computer that explains step-by-step its reasoning, its motivations, and the obstacles it is trying to overcome, and also analyzes emotional passages such as poetry. Such a computer is clearly not just parroting.

OBJECTION: A computer can't be kind, resourceful, beautiful, friendly, have initiative, have a sense of humor, tell right from wrong, make mistakes, fall in love, enjoy strawberries and cream, make someone fall in love with it, learn from experience, use words properly, be the subject of its own thought, have as much diversity of behavior as a man, or do something really new.
REBUTTAL: Why not? Although such a computer hasn't been built yet, it might be possible in the future.

OBJECTION: The computer never does anything original or surprising. It does only what it's told.
REBUTTAL: How do you know "original" human work isn't just grown from a seed implanted by teaching, or the effect of well-known general principles? And who says computers aren't surprising? The computer's correct answers are often surprisingly different from a human's rough guesses.

OBJECTION: Nerve cells can sense gradual increases in electrical activity—you can feel a "little tingle" or a "mild pain" or an "ouch"—whereas a computer's logic is only binary—either a "yes" or "no".
REBUTTAL: By using techniques such as "random numbers", you can make the computer imitate the flexible, probabilistic behavior of the nervous system well enough so that the Interrogator can't tell the difference.

OBJECTION: Life can't be reduced to rules. For example, if you have a traffic-light rule that says "stop when the light is red, and go when the light is green", what do you do when the light is broken, and both the red and the green appear simultaneously? Maybe you should have an extra rule saying in that case to stop. But some further difficulty may arise with that rule, and you'd have to create another rule. And so on. You can't invent enough rules to handle all cases. Since computers must be fed rules, they cannot handle all of life.
REBUTTAL: Although life's more than a simple set of rules, it might be the *consequences* of simple psychological laws of behavior, which the computer could be taught.

OBJECTION: Humans have extrasensory perception (ESP), and computers don't.
REBUTTAL: Maybe the computer's random-number generator could be hooked up to be affected by ESP. Or to prevent ESP from affecting the Imitation Game, put both the human witness and the computer in a telepathy-proof room.

To make the computer an intelligent creature, Turing suggested two possible ways to begin. One way would be to teach the computer abstract skills, such as chess. The other way would be to give the computer eyes, ears, and other sense organs, teach it how to speak English, and then educate it the same way you'd educate a somewhat handicapped child.

Four years later—on June 8, 1954—Turing was found dead in bed. According to the police, he died from potassium cyanide, self-administered. He'd been plating spoons with potassium cyanide in electrolysis experiments. His mother refuses to believe it was suicide, and hopes it was just an accident.

PLAY GAMES

TREES

In 1950, Claude Shannon proposed a way to make the computer win at checkers, chess, and other complicated games.

To understand his method, let's try to make the computer win a game of checkers. As in all checker tournaments, one player is called BLACK, and the other is called WHITE (even though his pieces are actually red). BLACK makes the first move. When a player can jump, he must. The game ends when one of the players can't move (either because he has no pieces or because his pieces are blocked).

To simplify the game, we'll play on a 4-by-4 board, instead of the traditional 8-by-8. Each player has two pieces instead of twelve.

Examine the diagram on the opposite page, which shows 63 possible positions. Position #1 is the initial position, from which black will move. The three arrows coming from position #1 represent the three legal moves he can choose from. Depending on which move he chooses, the board will wind up in position #2 or #3 or #4. Which move is best?

To find out, examine the consequences of each move:

If he moves to position #2, white will reply by moving to position #5 or #6 or #7.
If he moves to position #3, white will reply by moving to position #8 or #9 or #10.
If he moves to position #4, white will reply by moving to position #11 or #12 or #13.

The diagram shows all possible ways the game's first five moves could go. Throughout the diagram, w means white man, b means black man, Ẅ means white king, and B̈ means black king. The diagram is called a *tree*. (If you turn it upside down, it looks like the kind of tree that grows in the ground.) The arrows are called the tree's *branches*. The tree's *depth* is 5.

Which position should black choose: #2, #3, or #4? The wisdom of your answer depends on how deep you make the tree. In this particular game, a depth of 5 is satisfactory; but in 8-by-8 checkers or chess you might have to dig deeper. Theoretically, you should keep digging until you reach the end of the game; but such a tree might be too large to fit in your computer's memory.

For chess, Shannon estimated that a complete tree requires 10^{120} branches. Einstein estimated that the number of electrons in the universe is only 10^{110}. If Shannon and Einstein are both right, the tree can't fit in the universe!

Having constructed a tree of depth 5, look at the bottom positions (#42 through #63) and evaluate them, to see which positions look favorable for black. You should consider many factors: which player has control of the center of the board? which player can move the most without being jumped? and so on. But to keep matters simple, let's consider just one factor: which player has the most men? Consider a king to be worth $1\frac{1}{2}$ men.

Subtract the number of white men from the number of black men: the result of the evaluation is a number, which is called the position's *value*. If it's negative, black is losing; if it's positive, black is winning; if it's zero, the game is heading for a draw.

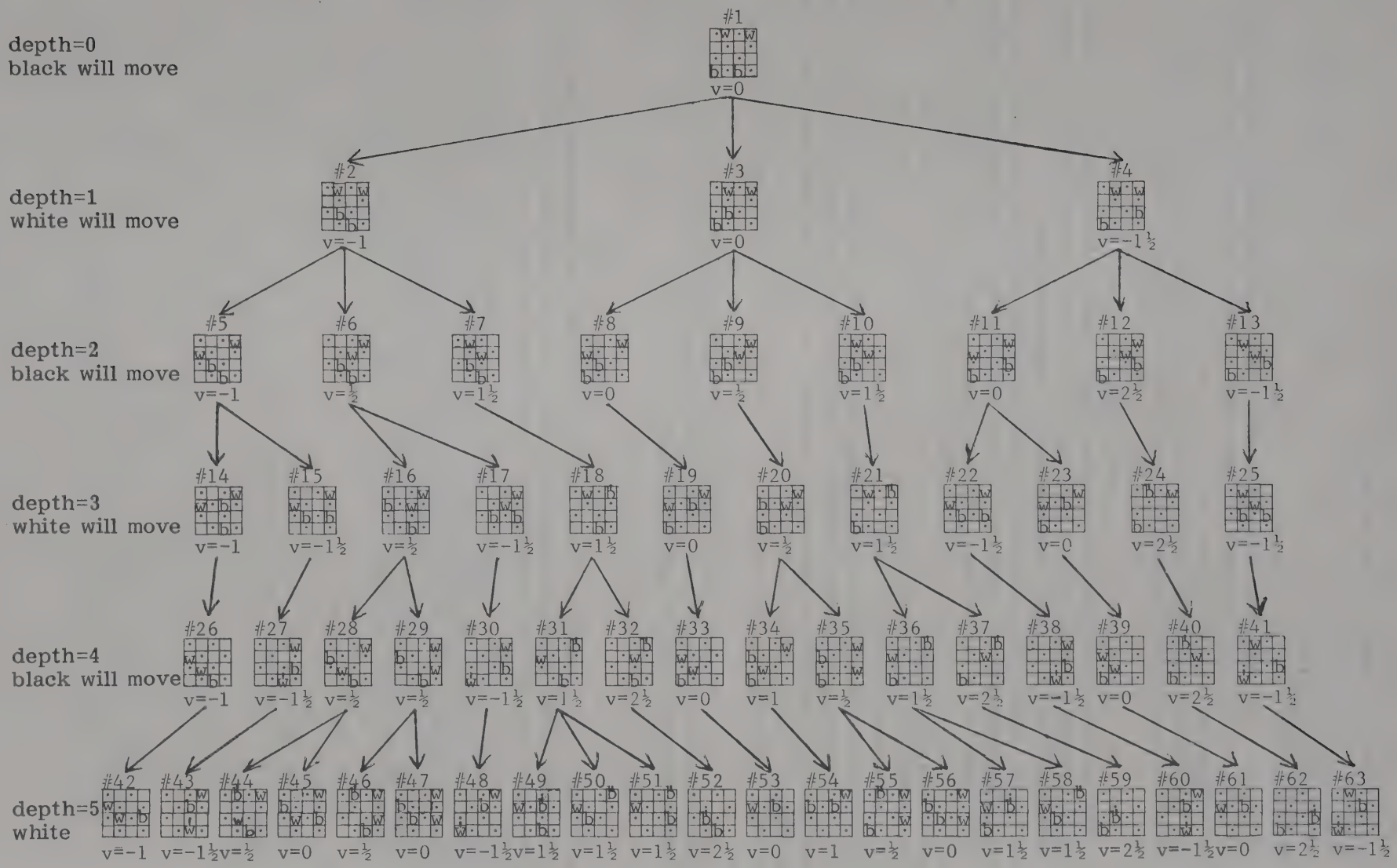
For example, consider position #42. Since black has one man and white has two, the value is 1 minus 2, which is -1. That's why I've written "v=-1" underneath that position. The value of each position at depth=5 is computed by that method.

For the positions at depth=4, use a different method. For example, here's how to find the value of position #92. That position has two possible outcomes: #46 and #47. Which outcome is more likely? Since the move will be made by black, and black's goal is to make the value large, he'll prefer to move to #46 instead of #47. Since the most likely outcome is #46, whose value is $\frac{1}{2}$, assign position #29 a value of $\frac{1}{2}$ also.

Here's the rule: to compute the value of a position at depth=4, find the *maximum* value of the positions it points to. (The value of position #29 is the maximum value of positions #46 and #47, which is $\frac{1}{2}$.)

To compute the value of a position at depth=3, find the *minimum* value of the positions it points to (since it's white's turn to move, and white wants to minimize). For example, the value of position #18 is the minimum value of positions #31 and #32, which is $1\frac{1}{2}$.

CHECKERS ON A 4-BY-4 BOARD



Compute the values for depth 2 by maximizing, and the values for depth 1 by minimizing. Finally, you get these results:

The value of position #2 is -1.
The value of position #3 is 0.
The value of position #4 is $-1\frac{1}{2}$.

Since black wants to maximize values, black should move to position #3. If white is also a good player, the game will probably gravitate toward position #53, a draw. If white is a poorer player, black will win.

That method of choosing the best move was proposed by Shannon. Since it makes heavy use of minimums and maximums, it's called the *minimax* method.

CHECKERS

After Shannon, the next person to become famous was Arthur Samuel. He spent a long time (twenty years, from 1947 to 1967) trying to make the computer win checkers. He used Shannon's minimax idea, but made many improvements.

His first spectacular success can in 1962, when his program won a game against Robert Nealey, a former Connecticut checkers champion. After the game, Nealey said, "The computer had to make several star moves in order to get the win....In the matter of the end game, I have not had such competition from any human being since 1954, when I lost my last game."

Later, the computer played six more games against Nealey. Nealey won one of them; the other five were draws.

In 1965 the computer played four games against W.F. Hellman, the World Champion. The games were played by mail. Under those conditions, Hellman won all four. But in a hastily played game where Hellman sat across the board from the computer, the result was a draw.

In 1967 the computer was beaten by the Pacific Coast Champion, K.D. Hanson, twice.

In short, the computer wins against most humans and draws against most experts, though it loses to the top champions. To bring the computer to that level of intelligence, Samuel improved Shannon's method in three ways:

When choosing among several moves, the computer analyzes the most promising ones more deeply.

After computing the value of a position (by examining the positions under it), the computer writes the value on a piece of tape. If the position recurs in another game, the computer looks at the tape instead of repeating the analysis.

To compute the value of a position, the computer examines many factors in addition to the number of pieces each player has. The computer combines the factors, to form combination-factors, and then combines the combination-factors to form a single value. The relative importance given to each factor is determined by "experience". Samuel experimented with two forms of experience: he had the computer play against itself, and also had it analyze 250,000 moves that occurred in checker championships.

MAXICOMPUTER CHESS

While Samuel was programming checkers, other programmers tried to write a similar program for chess. They had a hard time. In 1960 the best chess program that had been written was beaten by a ten-year-old kid who was a novice. The first decent chess program was written in 1967 by Richard Greenblatt and his friends at MIT: it actually won a game in a chess tournament. But in most tournaments, it lost. In 1970 and 1971, it lost every game in every tournament in entered.

In 1968, Atkins & Gorklen, undergraduates at Northwestern University, wrote a chess program. Inspired by their program, David Slate, a graduate student in physics there, wrote a chess program also. In 1969, Slate & Atkins combined the two programs, to form a better program, Chess 2.0. During the next several years, they continually improved the program; their most famous version was called Chess 4.7. Their program is playing chess against human experts— and winning! Their computer has scored several triumphs in tournaments designed for humans. In 1976, their computer was the winner of the class B section of the Paul Masson American Chess Championships (against the humans, it scored 5 wins, no losses). By winning that tournament, it achieved a U.S. Chess Federation score of 2210 and became a chess Master. Then it entered the Minnesota State Championship, to try to become the Minnesota State Champion, but lost (it scored 1 win, 3 losses, 1 tie).

In August 1968, an International Chess Master, David Levy, bet about $5,000 against several computerists. He bet that no computer would win a chess match against him in the next ten years. He won the bet: in August 1978, Chess 4.7 tried one last time to win a match against him, but lost (it scored 1 win, 3 losses, and 1 tie).

Slate & Atkins improved Chess 4.7, to form Chess 4.9, which is the world champion of computer chess. But though it's the world champion of computer chess, it's not necessarily the "best" program. It wins because it runs on a super-fast maxicomputer (manufactured by Control Data Corporation); other chess programs, written for slower computers, are at a disadvantage.

MINICOMPUTER CHESS

Almost as fast as Chess 4.9 is a program called Belle, written at Bell Telephone Laboratories. Belle runs on an unusual minicomputer that's specially wired to create trees quickly.

MICROCOMPUTER CHESS

Each of those programs— Chess 4.9 and Belle— requires an expensive CPU and lots of RAM. Is it possible to write a decent chess program using only a cheap CPU and very little RAM? Yes! In 1976, a Canadian named Peter Jennings wrote a program called Microchess 1.0; it ran on a $250 microcomputer (the KIM-1), which contained a 6502 CPU, no ROM, and only 1K of RAM! The program played decently, though not spectacularly.

Later, he improved the program, and called the improvement Microchess 1.5. It plays on the Radio Shack model 1 and the Apple. The version on the model 1 consumes 4K of RAM: 2K is for the logic, and the other 2K are just to make the picture of the chess board look pretty! You can get Microchess 1.5 for just $20 from your local Radio Shack store or Apple dealer.

In 1978, an amazing chess program was written by a husband-and-wife team: Dan and Kathe Sprachlin. They named the program "Sargon", to honor an ancient king. It ran on the Jupiter microcomputer, which contained an 8080 CPU and 16K RAM. It played much better than Microchess. When the Jupiter computer became obsolete, the Sprachlins rewrote the program, to make it run on the Radio Shack model 1 and the Apple. Then they improved the program; the improvement, called Sargon 2, runs on the Radio Shack, Apple, and several other computers. Sargon 2 is the best chess program for microcomputers. It plays almost as well as Chess 4.9 and Belle.

If you have a Radio Shack or Apple, you can buy Sargon 2 as a $30 tape or $35 disk from Hayden (50 Essex St., Rochelle Park, NJ 07662, 201-843-0550). If you have a Commodore VIC-20, you can buy Sargon 2 as a $40 ROM cartridge from Commodore (or your local Commodore dealer). When buying, make sure to ask for "Sargon 2", not just "Sargon".

Sargon 2 is fun. It lets you play against the computer and try to win. Probably, you're to stupid to win, and will lose to the computer. When it's your turn to move, you can— if you wish— press a special key, which makes the computer tutor you: the computer tells you how it would move, if it were in your position. You can follow the computer's advice, or ignore it. (Since your goal is to outsmart the computer, you should listen to the computer's advice; but instead of *following* the advice, you should try to devise an even cleverer move.)

After inventing Sargon 2, the Sprachlins invented Sargon 2.5, but it's *not* better than Sargon 2. Though Sargon 2.5 occasionally shows flashes of brilliance, it has a "fatal flaw": it never accepts a "draw". If the game is heading towards a draw, Sargon 2.5 avoids the draw by making unusual moves. Unfortunately, the unusual moves tend to make it lose. Sargon 2.5 is *not* available for Radio Shack, Apple, or Commodore; instead, it's part of a hand-held electronic chess game, which you can put in your purse.

Many companies manufacture hand-held electronic chess games. Some of the games even contain a tiny voice synthesizer, which lets the computer tell you its moves verbally. Some of the games even contain a mechanical arm, so that the computer will pick up the pieces and move them. Some of the games have touch-sensitive boards, so that you can indicate your move by just pushing the square you want to move from and the square you want to move to. For humor, some of the chess games have the computer make wisecracks about your style of playing.

CHOOSING A LEVEL

When you begin playing a top-notch computer game (such as Sargon 2), you must choose the "level" at which you want the computer to play. If you choose a low level, the computer will move quickly, without much forethought. If you choose a high level, the computer will play more carefully (and make better moves); to do that, the computer "looks ahead", by building a very large tree, which requires lots of time; and so you must wait a long time until the computer moves. If you choose a level that's very high, the computer will need *several hours* to compute its move.

If you buy Sargon 2, try playing at level 2, which is high enough to provide a challenging game yet low enough to make the game progress quickly.

WHY A COMPUTER?

Playing against the computer is more interesting than playing against a human.

When you play against a human friend, you must wait a long time for your friend to move. When you play against Sargon 2 at level 2, the computer moves almost immediately. So you can play several games against the computer (and learn a lot from them) in the same amount of time you'd need to play just *one* game against a human. So by playing against the computer, you gain experience faster than by playing against a human. Bobby Fisher, the world chess champion, now plays *only* against computers; he refuses to play against humans.

The computer is kinder than a human. If you make a bad move, the computer lets you "take it back" and try again. If you seem to be losing, the computer lets you restart the whole game. The computer— unlike a human— has infinite patience and no ego. Playing against the computer is less threatening than playing against a human.

If you have a computer, you don't have to worry about finding an opponent who's "at your level"; when you play against the computer, just tell the computer at what level you want it to play. The computer will act about as smart as you wish.

OTHELLO

Chess and checkers are both played on a checkerboard. Another game that's played on a checkerboard is "Othello". It uses checkers, but each checker has two sides: one side is white; the flip side is black.

When the game begins, only four checkers are on the board: two of them have their white side showing, and the other two checkers show black.

The game is for two players. One is called the white player, and the other is called the black player.

For example, suppose you're the white player. On your turn, you put an extra checker onto the board, so that the checker shows white. You must position the checker so that it and a previously placed white checker surround some black checkers. Then you flip all the surrounded black checkers, so that they become white.

Similarly, on his turn, the black player puts a black checker onto the board, so that some of your white checkers are surrounded by black, and he flips all those white checkers, so that they become black.

The game ends when the board is entirely filled with checkers. If most of the checkers are white, the white player wins; otherwise, black wins.

The game is tricky, because the definition of "surrounded checkers" is strange, and because you can't easily figure out who's winning. At first glance, you'd think that if most of the checkers on the checkerboard are white, white is ahead; but at the end of the game, the situation can change drastically. For example, the black player might place a black checker in such a way that most of the white checkers become black. So you must guard against dangerous positions. During the early parts of the game, the white checkers' *positions* are more important than their *quantity*.

The game begin centuries ago in England, where it was called "Reversi". It resembled the Japanese game called "Go". About 1975, it was marketed in the United States as a board game, under the name "Othello" (which is trademarked by Gabriel Industries). Programmers tried to make the computer imitate the game and win.

After writing Sargon 2 (the award-winning chess program), Dan and Kathe Sprachlin turned their attention to Othello, and wrote an award-winning Othello program, called "Reversal". It plays Othello better than any other program ever invented. Like Sargon 2, it's published by Hayden, runs on the Apple, allows several levels of play, costs $35 on disk, and lets you press a "tutoring" button whenever you want the computer to give you advice on how to reply. For added humor, each checker shows a frown or a smile. For example, if the white checkers outnumber the black, the white checkers wear smiles, and the black checkers wear frowns; the smiles and frowns grow bigger, as white's lead over black increases. And whenever a checker is added to the board or flipped, the computer plays a musical fanfare.

BACKGAMMON

Backgammon is a game played with dice. It requires both luck and skill. For many years, the world champion backgammon player was a human. But recently, he was beaten by a computer, in a thorough match.

The human was a poor loser: he blamed it on "bad luck". He refuses to admit that the computer has more skill than he. Nevertheless, the computer is now the world champion.

ADVENTURE

"Adventure" is a game where you hunt for treasure. The original version of Adventure was written for the PDP-10 maxicomputer at Stanford University's Artificial Intelligence Laboratory.

When you run the program, the computer says you're at the end of a road, near a shack, and says the computer will act as your body and understand any two-word command. Then it waits for you to give it a command. You can tell it to GO NORTH or GO FORWARD or— if you're going along a stream— you can say FOLLOW STREAM or GO DOWNSTREAM. The first time you play this game, you feel lost— the game is an adventure. As you wander in whatever direction you please, the computer says you're going through forests, across streams, over hills, etc. Finally, after much aimless wandering, you'll eventually see a stream which, if you follow it, leads you to a mysterious iron grate. If you try to BREAK GRATE, the computer says you're not strong enough. If you try to OPEN GRATE, the computer says you have no keys. Your curiousity aroused, you start feeling frustrated. Then the computer offers to give you a hint— but the hint will cost you several points. If you acquiesce, the computer will

give you this hint: find the keys!

To find the keys, the average idiot who plays this game tries wandering through the forests and valleys again. But if you're smart, you'll remember that at the beginning of the adventure you were next to a shack; so you go back to the shack, walk inside, and lo and behold, you find the keys! So you trek back to the iron grate, and use the keys to get in. And you think— aha! — you've almost solved whatever the problem is.

But actually, you've just begun! The grate leads you into a cave that contains 130 rooms, which form a big three-dimensional maze. Somewhere in the maze lie 15 buried treasures. But as you walk through the maze, you tend to keep forgetting where you are and where you've come from; you can waste lots of time just walking in circles, without realizing it! And to make matters worse, the cave contains many dangers, such as trap doors (if you fall in, you break every bone in your body!) and trolls and snakes, which you must ward off by using various devices which you must find in the cave's rooms or even back at the shack. Yes, you might have to trek all the way back to the shack again!

Finally, after dodging all the evil things in the cave, you finally get to the treasures. You pick them up and start walking away with them. But then you hear footsteps behind you. And the pirates steal your treasure! Then you must chase the pirates.

If you manage to retain your treasures and your life and get out of the cave, that doesn't mean you've won. The nasty computer keeps score of how *well* you do in retrieving the treasures. The maximum possible score is 350. After you've played this game many times and learned how to duck all your adversaries quickly, you'll find your score is still only 349, and you'll wonder what you did wrong that cost you 1 point. The answer is: at one point in the adventure, you must borrow magazines from one of the cave's rooms; to get the extra point, you must return them!

The game is a true adventure, because as you wander through forests and the rooms in the cave, the computer tells what you see, but you don't know whether what you see is important. For example, when you walk into a room, the computer might say the room contains a small cage. That's all it says. You must guess whether the cage has any significance, what that significance would be, and what you should do to the cage, if anything. Should you pick it up? Try to break it? Kiss it? Carry it? Try anything you like— give any command to your computer-body that you wish— and see what happens.

Although Adventure was originally written for a PDP-10 maxicomputer, you can get an exact imitation for your Apple. The Apple disk costs $30 and is published by Microsoft (10700 Northup Way, Bellevue, WA 98004, 206-828-8080).

A quite similar version is available for CP/M computers. It costs $25, is bilingual (you play in either English or French!), and is published by Creative Computing (39 E. Hanover Ave., Morris Plains, NJ 07950, 201-540-0445, 800-631-8112).

Each of those versions of Adventure displays beautiful prose but no graphics. If you want graphics— or want your adventure to occur elsewhere than in a cave— get one of the hundreds of variations that other programmers have written. You can even get a version where, instead of exploring a cave, you explore a brothel: it's called *Softporn Adventure* and is rated X. It comes as a $30 Apple disk from Sierra On-Line (Sierra On-Line Building, Coarsegold, CA 93614, 209-683-6858).

Sierra On-Line has developed an even fancier version of Adventure. The fancier version is called *Time Zone*. The Time Zone program is so long that it fills *both* sides of *6* Apple disks; that's 12 sides altogether! In fact, the game is so long that nobody's ever finished playing it! Here's how to play....

You use a computerized "time machine", which transports you to 9 times (400 million B.C., 10000 B.C., 50 B.C., 1000 A.D., 1400, 1700, 1982, 2082, and 4082) and 8 locations (North America, South America, Europe, Africa, Asia, Australia, Antarctica, and Outer Space). Wherever you go, your screen shows a high-resolution color picture of where you are; for example, if you choose "approximately 1400", Christopher Columbus will welcome you aboard his ship. Altogether, the game contains over 1400 pictures! You travel through history, searching for clues that help you win.

Time Zone is well-researched: it's historically accurate. And it doesn't let you cheat. For example, in the year 2082, you find a book of matches; using your time machine, you can carry the matches back to 1982 but not back to 1700— because matches weren't invented until 1800.

Living through history isn't easy. My friend Jonathan Rotenberg, who's played the game, says:

I've been killed dozens of times. I've been assassinated by Brazilian terrorists, karate-chopped by a Brazilian monk, eaten by a tyrannosaur, crushed in an Andes avalanche, stampeded by a buffalo, overcome by Antarctic frostbite, and harpooned by Mayan fishermen.

And you see it all in color!

Time Zone costs $100. For more of Jonathan's comments about it, read his review in the May/June 1982 issue of *Computer Update* (published by the Boston Computer Society, 3 Center Plaza, Boston, MA 02108, 617-367-8080).

SPORTS

The best basketball game is the one sold by Atari, for Atari computers. The best soccer game is the one sold by Texas Instruments, for the TI-99/4. Each of those games comes as a $30 ROM cartridge and includes color graphics and sounds. In basketball, you hear the ball dribble; in soccer, you hear the crowds cheer.

The most realistic road race is *International Gran Prix*. You must even switch gears, and avoid skidding and fishtailing. Your dashboard shows speed, distance, fuel, time, engine RPM, and many fancier indicators. You can choose your own race: pick the race course, the weather conditions, the amount of fuel, and the number of laps. If you're lazy, you can pick a car that has automatic transmission. Afraid of going too fast? Then turn on the cruise control. The game includes sounds: you hear the motor revving up; if you drive poorly, you end in a crash. The $30 Apple disk is published by Riverbank Software (Smith's Landing Road, P.O. Box 128, Denton, MD 21629, 301-479-1312, 800-327-9191 ext. 244).

To become an Olympic star, buy *Olympic Decathlon*, which puts you through all ten events of the decathlon: the 100-meter dash, long jump, shot-put, high jump, 400-meter dash, 110-meter hurdles, discus throw, pole vault, javelin throw, and 1500-meter run. You can play alone, or play against your friends. As your fingers move across the keys, your television screen shows your body moving through the ten events. This program won Creative Computing's award for being the "best computer game of the year". The programmer, Tim Smith, spent 9 months, full-time, writing it for the Radio Shack model 1; then he spent 9 more months, full-time, rewriting it for the Apple; then he rewrote it *again*, for the IBM Personal Computer. The IBM version is fancier than the Apple version, which is fancier than the Radio Shack version. Each version comes as a $25 disk and is published by Microsoft. Get it from your local Radio Shack, Apple, or IBM dealer.

SPACE WAR

In the 1960's, many kids watched a television show called *Star Trek*, in which a spaceship (called the Enterprise) and its crew (Captain Kirk, Mr. Spock, McCoy, and underlings) killed nasty creatures called Klingons, by using torpedoes called phasers. Their ship traveled to different sectors of the universe, by using "impulse engines" (which were slow) and "warp engines" (which made the spaceship go even faster than the speed of light, but which required dilithium crystals). Many programmers wrote games throwing you into the action: you travel from sector to sector, find the nasty Klingons, and kill them. The best Star Trek game is the one sold by Atari, for Atari computers: it's called *Star Raiders*, costs $40, and boasts three-dimensional color graphics and incredible whoosing sounds.

If you visit a game arcade, you'll see— beside the pinball machines— a computerized game called *Space Invaders*, in which you shoot aliens who are hovering overhead and dropping bombs on you. The game was invented in Japan; the name "Space Invaders" is copyrighted by Taito Corp. It's the most popular computer game ever invented: 2½ million copies of it have been sold. A popular variation is *Galaxian*, in which the aliens dive down to get you, kamikazi-style. An even newer variation is *Phoenix*, where the dive-bombing aliens suddenly turn into gigantic phoenix birds. Other popular shoot-em-up games are *Defender* (where you fly across a landscape of goblins), *Asteroids* (where you shoot asteroids that are coming at you), *Missile Command* (where you use anti-ballistic missiles to protect cities in a nuclear war), *Pacman* (where you run through a maze while you eat everything in sight but avoiding ghosts who are trying to eat *you*), and *Donkey Kong* (where you must climb ladders to save the princess from King Kong).

Each of those games has been imitated on microcomputers.

HANDLE ENGLISH

FAMOUS ATTEMPTS

Here are some famous attempts to make the computer understand plain English....

BASEBALL (1961)

In 1961 at MIT, programmers made the computer answer questions about baseball. In the computer's memory, they stored the month, day, place, teams, and scores of each game in the American League for one year. They programmed the computer so that *you can type your question in ordinary English*. The computer analyzes your question's grammar and prints the correct answer.

Here are examples of questions the computer can analyze and answer correctly:

Who did the Red Sox lose to on July 5?
Who beat the Yankees on July 4?
How many games did the Yankees play in July?
Where did each team play in July?
In how many places did each team play in July?
Did every team play at least once in each park in each month?

To get an answer, the computer turns your questions into equations:

Question	Equations
Where did the Red Sox play on July 7?	place = ? team = Red Sox month = July day = 7
What teams won 10 games in July?	team (winning) = ? game (number of) = 10 month = July
On how many days in July did eight teams play?	day (number of) = ? month = July team (number of) = 8

To do that, the computer uses this table:

Word in your question	Equation
where	place = ?
Red Sox	team = Red Sox
July	month = July
who	team = ?
team	team =

If your question mentions *Boston*, you might mean either "place = Boston" or "team = Red Sox"; the computer analyzes your question to determine which equation to form. Words such as *the*, *did*, and *play* are ignored.

After forming the equations, the computer hunts through its memory, to find the games that solve the equations. If an equation says "number of", the computer counts. If an equation says "winning", the computer compares the scores of opposing teams.

The programmers were Bert Green, Alice Wolf, Carol Chomsky, and Kenneth Laughery.

EXCEPTIONS (1964)

In 1964, MIT awarded a Ph.D. to Bertram Raphael, for making the computer understand generalizations and exceptions. Here are conversations between a human and Raphael's computer:

CONVERSATION 1

HUMAN: *There are 5 fingers on every hand. There are two hands on a person. A boy is a person. Tom is a boy. Dick is a boy. Harry is a boy. Tom has nine fingers. Dick has one hand. How many fingers does Tom have?*
COMPUTER: *The answer is 9.*

HUMAN: *How many fingers does Dick have?*
COMPUTER: *The answer is 5.*

HUMAN: *How many fingers does Harry have?*
COMPUTER: *The answer is 10.*

CONVERSATION 2

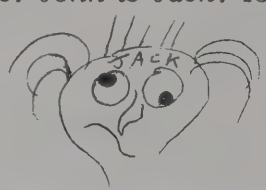
HUMAN: *Every keypunch-operator is a girl. Any girl is an example of a person. Is a keypunch-operator a person?*
COMPUTER: *Yes.*

HUMAN: *Is a person a girl?*
COMPUTER: *Sometimes.*

HUMAN: *Is a monkey a keypunch-operator?*
COMPUTER: *Insufficient information.*

CONVERSATION 3

HUMAN: *The man is a jerk. Jack is a dope. John is Jack. Is John a dope?*
COMPUTER: *Yes.*



HUMAN: *Is the man a dope?*
COMPUTER: *Yes.*

HUMAN: *Jim is a man. Is the man a dope?*
COMPUTER: *Which man?*

CONVERSATION 4

HUMAN: *The telephone is just to the right of the book. The telephone is just to the left of the pad. Is the pad just to the right of the book?*
COMPUTER: *No.*

HUMAN: *The pad is to the right of the telephone.*
COMPUTER: *The above statement is already known.*

HUMAN: *The pad is to the left of the telephone.*
COMPUTER: *The above statement is impossible.*

HUMAN: *The ash-tray is to the left of the book. The pencil is to the left of the pad. The paper is just to the right of the telephone. Where is the pad?*
COMPUTER: *Just to the right of the telephone. Somewhere to the right of the pencil.*

HUMAN: *What is the position of the pad?*
COMPUTER: *The left-to-right order is as follows: ash-tray, book, telephone, pad, paper. To further specify the positions, you must indicate where the pencil is with respect to the ash-tray.*

HUMAN: *The book is just to the right of the ash-tray. What is the position of the pad?*
COMPUTER: *The left-to-right order is: pencil, ash-tray, book, telephone, pad, paper.*

HUMAN: *A telephone is an audio-transducer. A diaphragm is part of an audio-transducer. Where is the diaphragm?*
COMPUTER: *Just to the left of the pad. Just to the right of the book. Somewhere to the left of the paper.*

When you begin running that program, the computer already understands words such as *a*, *the*, *is*, *every*, *has*, *owns*, *part*, and *left*. The program has two major limitations: your grammar must be simple, and the computer does only a few forms of reasoning.

WHAT'S A STORY PROBLEM?

When you were in school, your teacher told you a story that ended with a mathematical question. For example:

Dick gave Jane 5 pounds of cocaine. Jane snorted 3 pounds, then died. When Dick came to kiss Jane's dead body, how many pounds of cocaine did he find left?

In that problem, the last word is: *left*. That means: subtract. So the correct answer is 5 pounds minus 3 pounds, which is 2 pounds— unless Jane's friends got there before Dick.

Can the computer solve problems like that? Here's the most famous attempt....

ARITHMETIC & ALGEBRA (1964)

MIT awarded a Ph.D. to Daniel Bobrow, for programming the computer to solve story problems involving arithmetic and algebra.

Let's see how the computer solves this problem:

If the number of customers Tom gets is twice the square of 20 per cent of the number of advertisements he runs, and the number of advertisements he runs is 45, what is the number of customers Tom gets?

To begin, the computer replaces *twice* by *2 times*, replaces *square of* by *square*, and replaces *per cent* by *percent*. Then the computer separates the sentence into smaller sentences:

The number of customers Tom gets is 2 times the square 20 percent of the number of advertisements he runs. The number of advertisements he runs is 45. What is the number of customers Tom gets?

The computer turns each sentence into an equation:

*number of customers Tom gets = 2 * (.20 * number of advertisements he runs)↑2*
number of advertisements he runs = 45
X = number of customers Tom gets

The computer solves the equations and prints the answer as a complete sentence:

The number of customers Tom gets is 162.

Here's a harder problem:

The sum of Lois's share of some money and Bob's share is $4.50. Lois's share is twice Bob's. Find Bob's and Lois's share.

Applying the same method, the computer turns the problem into these equations:

Lois's share of some money + Bob's share = 4.50 dollars
*Lois's share = 2 * Bob's*
X = Bob's
Y = Lois's share

The computer tries to solve the equations but fails. So it assumes "Lois's share" is the same as "Lois's share of some money", and "Bob's" is the same as "Bob's share". Now it has six equations:

Original equations
Lois's share of some money + Bob's share = 4.50 dollars
*Lois's share = 2 * Bob's*
X = Bob's
Y = Lois's share

Assumptions
Lois's share = Lois's share of some money
Bob's = Bob's share

It solves them and prints:

Bob's is 1.50 dollars.
Lois's share is 3 dollars.

Here's an even harder one:

The distance from New York to Los Angeles is 3000 miles. If the average speed of a jet plane is 600 miles per hour, find the time it takes to travel from New York to Los Angeles by jet.

The resulting equations are:

*distance from New York to Los Angeles = 3000 * miles*
*average speed of jet plane = (600 * miles)/(1 * hours)*
X = time it takes to travel from New York to Los Angeles by jet

The computer is unable to solve them. But in its memory it finds the formula "distance = speed * times". It winds up with 7 equations:

Original equations
*distance from New York to Los Angeles = 3000 * miles*
*average speed of jet plane = (600 * miles)/(1 * hours)*
X = time it takes to travel from New York to Los Angeles by jet

Formula
*distance = speed * time*

Assumptions
distance = distance from New York to Los Angeles
speed = average speed of jet plane
time = time it takes to travel from New York to Los Angeles by jet

It solves them and prints:

The time it takes to travel from New York to Los Angeles by jet is 5 hours.

The computer can handle problems about age:

Mary is twice as old as Ann was when Mary was as old as Ann is now. If Mary is 24 years old, how old is Ann?

To begin, the computer replaces *twice* by *2 times*, and replaces *how old* by *what*. When the computer sees the words *as old as*, it knows the problem has to do with ages, so it breaks the problem into these sentences:

Mary's age is 2 times Ann's age X years ago. X years ago Mary's age is Ann's age now. Mary's age is 24. What is Ann's age?

The resulting equations are:

*Mary's age = 2 * (Ann's age - X)*
Mary's age - X = Ann's age
Mary's age = 24
Y = Ann's age

The computer prints:

Ann's age is 18.

If the situation appears hopeless, the computer asks you for help. In the following conversation, the human poses a problem, and the computer begs for a hint:

HUMAN: *The gross weight of a ship is 20000 tons. If its net weight is 15000 tons, what is the weight of the ship's cargo?*
COMPUTER: *Do you know any more relationships among these variables: gross weight of ship, tons, its net weight, weight of ship's cargo?*

HUMAN: *Yes.*
COMPUTER: *Tell me.*

HUMAN: *The weight of a ship's cargo is the difference between the gross weight and the net weight.*
COMPUTER: *The weight of the ship's cargo is 5000 tons.*

The program has some limitations. It cannot solve quadratic equations. If the computer tries to apply the formula "distance = speed * time" to a problem involving *two* distances, it forgets which distance is which. The computer's vocabulary is limited: it doesn't realize that *how many* means *what is the number of*, and *how far is* means *what is the distance to*. If the human types *Tom has 2 apples, 3 bananas, and 4 pears*, the comma before the *and* makes the computer split the sentence into two wrong "sentences":

Tom has 2 apples, 3 bananas.
4 pears.

If the human mentions *the number of times John went to the movies*, the computer thinks *times* means multiplication, and tries to multiply *number of* by *John went to the movies*. For problems about age, the characters must be named Mary, Ann, Bill, Father, or Uncle, unless you diddle with the computer's memory.

ENCYCLOPEDIA (1964-1966)

In 1964, Simmons, Klein, and McConlogue (at the Systems Development Corporation) fed a child's encyclopedia into the computer. If you type *What do worms eat?* the computer hunts through the encyclopedia, to find sentences mentioning both *worms* and *eat*. (To hasten the hunt, it uses an index produced by another program.) It finds two sentences:

Birds eat worms on the grass.
Most worms usually eat grass.

After analyzing the grammar of your question and those two sentences, the computer realizes the first sentence is irrelevant, and prints just the correct answer:

Most worms usually eat grass.

In 1965, the program's abilities were extended, so that if you type *What animals live longer than men?* the computer searches through the encyclopedia for related words:

Word in sentence	Related words searched
animals	mammals, reptiles, fish
live	age
longer	older, ancient
men	person, people, women

Here's an example of the program's operation in 1966. The human creates some encyclopedia material about horses, and then retrieves it:

HUMAN: *Enter horse=noun, is=isverb, animal=noun, live=verb, on=prep, farm=noun, strong=adj, pull=verb, wagon=noun, most=adj, are=isverb, eat=verb, hay=noun, all=adj, red=adj, many=adj, have=haveverb, long=adj, mane=noun, some=adj, person=noun, race=verb.*
COMPUTER: *Entered in dictionary.*

HUMAN: *Plurals horse=horses, animal=animals, farm=farms, wagon=wagons, mane=manes, person=people.*
COMPUTER: *Ok.*

HUMAN: *Start.*
COMPUTER: *Ready.*

HUMAN: *A horse is an animal. Horses live on a farm. Strong horses pull wagons. Most horses are strong. Horses eat hay. All horses are red. Many horses have long manes. Some people race horses.*
COMPUTER: *Learned.*

HUMAN: *Forget "all horses are red".*
COMPUTER: *Forgotten.*

HUMAN: *What is a horse?*
COMPUTER: *Horse is an animal.*

HUMAN: *What do horses eat?*
COMPUTER: *Horses eat hay.*

HUMAN: *What do horses do?*
COMPUTER: *Horses live on a farm and eat hay. Strong horses pull wagons.*

HUMAN: *What do strong horses do?*
COMPUTER: *Strong horses pull wagons.*

HUMAN: *Do many horses have long manes?*
COMPUTER: *Yes.*

HUMAN: *Do all horses have long manes?*
COMPUTER: *To my knowledge, no.*

HUMAN: *Describe horses?*
COMPUTER: *Horses are an animal. Most horses are strong. Many horses have long manes. Horses live on a farm and eat hay. Strong horses pull wagons. Some people race horses.*

When the computer's answer consists of several sentences, it prints the most relevant sentence first, and the others in decreasing order of relevance.

THE MILITARY (1966)

Thompson and Craig (at General Electric) made the computer answer military questions such as:

Has the 25th batallion arrived in Texas since 3 P.M.?
Is the 100th scheduled to arrive at Ft. Lewis before the 200th leaves Ft. Lewis?

COMMON CONCEPTS (1966)

Carnegie-Mellon University awarded a Ph.D. to M. Ross Quillian, for making the computer compare words. If you type two words, the computer finds a concept they have in common:

HUMAN: *cry, comfort*
COMPUTER: *Cry is among other things to make a SAD sound. To comfort can be to make something less SAD.*

HUMAN: *plant, man*
COMPUTER: *Plant is not a ANIMAL structure. Man is ANIMAL. To plant is for a PERSON someone to put something into earth. Man is PERSON.*

HUMAN: *plant, industry*
COMPUTER: *Plant is apparatus which person use for process in INDUSTRY.*

PSYCHOTHERAPY & GEOMETRY & GERMAN (1966-1967)

In 1966 at MIT, Joseph Weizenbaum wrote the Eliza program, which makes the computer act as a psychotherapist.

In 1967, he changed the program, to make it handle keywords such as *also*, *call*, *compute*, *define*, *is*, *let*, *of*, *thanks*, *where*, and the symbol =. In the following conversation, the human, teaches the computer how to find the area of a circle, a sphere, and a square, and then asks the computer questions about geometry. The human also teaches the computer how to speak German—

English	German
I say	ich sage
I mean	meine ich
what is	wieviel ist
squared	quadrat

and gradually starts speaking to the computer in German.

HUMAN: *The area of a circle is pi times the radius of the circle squared. The area of a sphere is pi times the diameter of the sphere squared. The area of a square is the side of the square squared. The diameter of an object is twice the radius of the object. The radius of a ball is 10. What is the area of the ball?*
COMPUTER: *The assumption that ball is circle is consistent— shall I go on?*
HUMAN: *No.*
COMPUTER: *The assumption that ball is sphere and object is consistent— shall I go on?*
HUMAN: *Yes.*
COMPUTER: *It's 1256.6359.*

HUMAN: *What is the diameter of the ball?*
COMPUTER: *It's 20.*

HUMAN: *The radius of a dish is 10. What's the area of the dish?*
COMPUTER: *The assumption that dish is circle is consistent— shall I go on?*
HUMAN: *Yes.*
COMPUTER: *It's 314.15899.*

HUMAN: *The radius of a globe is 10. A globe is a sphere. A sphere is an object. What is the area of the globe?*
COMPUTER: *It's 1256.6359.*

HUMAN: *The side of a plate is 10. What is the area of the plate?*
COMPUTER: *The assumption that plate is square is consistent— shall I go on?*
HUMAN: *Yes.*
COMPUTER: *It's 100.*

HUMAN: *When I say board I mean plate. What is 3 times the side of the board?*
COMPUTER: *It's 30.*

HUMAN: *Please call that x. Let f(z) be z squared minus z. Compute f(x).*
COMPUTER: *It's 870.*

HUMAN: *When I say ich sage I mean I say. When ich sage meine ich I mean I mean. Wenn ich sage wieviel ist meine ich what is. Wenn ich sage quadrat meine ich squared. Nun, sagen sie mir bitte, wieviel is x quadrat.*
COMPUTER: *I didn't understand you.*
HUMAN: *Ja, tut mir leid, ich sollte fragen wieviel IST x quadrat.*
COMPUTER: *It's 900.*

CALCULUS (1969)

At MIT, Charniak made the computer solve story problems involving calculus.

HUMAN: *Water is flowing into a conical filter at the rate of 15 cubic inches per second. If the radius of the base of the filter is 5 inches and the altitude is 10 inches, find the rate at which the water level is rising when the volume is 100 cubic inches.*
COMPUTER: *The answer is .53132943 * in * sec↑-1 * pi↑-.33333332.*

PROBABILITY (1971)

Rensselaer Polytechnic Institute awarded a Ph.D. to J.P. Gelb, for making the computer solve story problems involving probability.

HUMAN: *From a zorch containing 4 ferd and 3 brakky and 5 chartreuse werfels, 3 are drawn. What is the probability that 2 are chartreuse and the other brakky?*
COMPUTER: *Replacement involved?*
HUMAN: *No.*
COMPUTER: *3/22 (or .1363636).*

REPLACE PEOPLE

BARTENDERS

Many bar owners don't trust the bartenders they hire. They claim the bartenders give too many free drinks to friends, steal money from the till, and put too much or too little liquor in the drinks.

To solve the problem, many bars now contain a computer that mixes and pours drinks. The computer mixes accurately. Although the computer is run by the bartender, the computer keeps an accurate record of how many drinks it makes, so there is little chance for cheating. The computer also keeps track of the inventory.

The computers are manufactured and sold by National Cash Register Co. (Dayton, Ohio), Bar Boy Inc. (San Diego, California), Electronic Dispensers International (Concord, California), and Anker-Werke (Germany). Prices range from $600 to $15000. Holiday Inns has been developing its own model.

DOCTORS

This article (abridged) appeared in *The Times*:

DIAGNOSIS BY COMPUTER MORE ACCURATE
BUT DOCTORS STILL NEEDED

A medical diagnostic system designed at Leeds University has proved more accurate than doctors in assessing the most likely cause of acute abdominal pain among patients admitted to the university's department of surgery.

Last year 304 such patients were admitted to the unit, and the computer's diagnosis proved correct in 92 percent of the cases, compared with 80 percent accuracy by the most senior doctor to see each case.

After each patient had been seen by the doctor and examined, the doctor's findings were passed on to a technician, who translated them into language used by the computer. The computer would list the likely diagnoses in order of probability. If the computer and the doctor in charge of the case disagreed, the computer would on request suggest further investigations that might be useful.

In the year-long trial the computer's diagnoses proved correct in 279 cases. In fifteen it was wrong, in eight the patient's condition was not included in the diseases considered by the computer, and in two no computer diagnosis was made because the doctors concerned with the case disagreed about the findings.

Whereas the computer advised an operation on six occasions when it would have proved unnecessary, in practice 30 such operations were carried out on the basis of the surgeon's own judgment. The computer accurately classified 84 of the 85 patients with appendicitis, compared with 75 by the doctors, and its suggestion that no operation was necessary proved correct on 136 out of 137 occasions.

The computer is reliable only if accurate data are fed into it on the basis of the doctor's interrogation and examination of the patient.

This article (abridged) is from the magazine *Computers and Automation*:

COMPUTER POISON CONTROL CENTER
OPENED BY CHILDREN'S MERCY HOSPITAL

A computerized poison control center, designed to help save children's lives by quickly identifying poisons they swallow, has been opened by The Children's Mercy Hospital, Kansas City, Mo. The computer stores information on drugs, household products, and chemicals that children may find and swallow. The computer is programmed to accept an inquiry—for example, the name of a household cleaning product—and to return to the terminal within four seconds detailed information on the poison, including symptoms and suggested treatment. The system does not replace clinical judgment, but does save valuable time in locating the requested poison.

In an experiment at another hospital, each patient was given a computer terminal and could use it to communicate with the computer about his illness. Afterwards, most of the patients said they'd rather deal with the computer than with human doctors.

This report is by James Martin and Adrian Norman:

The Los Angeles County General Hospital Shock Research Unit is concerned with circulatory shock, which results in low blood pressure, low blood flow, and an extremely unstable circulatory system. The proportion of circulatory-shock patients who die is large. A computer was installed to improve the monitoring and care of these patients. A patient in this ward has many instruments attached to him: six thermistors measure the temperature at various parts of his body; pressures are measured in the circulatory system; catheters are inserted into an artery and a vein; and a urinometer measures liquid waste. The instruments are all directly connected to the computer, and other details about the patient are given to the machine by the nurse. The computer prints out a log about the patient's condition, and if any measurement falls outside critical limits set by the doctor for that patient, the computer notifies the ward staff.

TUTORS

Judith Harris, John Swets, and Wallace Feurzeig made the computer run a logic game. The computer thinks of a letter of the alphabet; your goal is to discover which letter it is, as quickly as possible. You can ask the computer questions about the letter's shape.

In this example, the human (in *italics*) begins by asking whether the letter has any curves:

Curves?
One.
Loose ends?
Two.
Obliques?
Zero.
C.
You don't have enough information yet to get the right answer. How do you know it isn't J, for example?
J.
You don't have enough information yet. How do you know it isn't C, for example?
Horizontals?
Zero. That was a wasted question. It did not eliminate any possibilities.
Verticals?
Zero. Good question.
G.
Wrong. Your guess is inconsistent with the information you've been given. You must have forgotten that the right letter hasn't any HORIZONTAL LINES. The letter you guessed can't be right—it has too many LOOSE ENDS.
Curves?
The answer is still: one.
X.
Wrong. Your guess is inconsistent with the information you've been given. I think you only made that guess to see what I would say.
C.
Yes, you've done it. Good work. If you want to play again, start right in.

Notice that the human can input either a guess (followed by a period) or a question (followed by a question mark). Possible guesses are *A*, *B*, *C*, *D*, etc. Possible questions are *Curves*, *Loose ends*, etc.; the human is given a list of possible questions before he runs the program.

That logic game was modified, to form a lesson about medicine. The medical case was designed by Preston Munter, of the Harvard Health Center; the programming was done by John Swets, Wallace Feurzeig, and Myra Breen. The object of the "game" is to diagnose an illness correctly, by performing sufficient tests. The human can input either a diagnosis (*Measles.*) or a request for a test (*X-ray?*). Before running the program, the human is given a list of 35 possible diagnoses and 40 possible tests.

Here are excerpts from a long RUN:

```
Please type your name, the date, and a slash.
Myra Breen, August 1963/
It is mid-December. Late this afternoon, a patient was admitted to the hospital.
He is a well-developed, well-nourished, 34-year-old Negro coal miner. He was brought
in by his wife, who reports that he has had "the grippe" for about 4 days. During
the past 3 days she has given him fluids and aspirin every 6 hours with no sign of
improvement. Today he suddenly became much worse. For the past 2 days he had
complained of feeling very warm and of pain in his belly. This morning he had a
severe shaking chill that was so bad his teeth rattled.
...
Begin the physical examination as soon as you are ready.
Physical exam?
Fine. What in particular would you like to know?
Chest exam?
It is common practice in the medical profession to begin an examination by checking
the patient's general appearance.
General appearance?
Acutely prostrated, lying in bed on his right side. Looks acutely ill, sweaty,
apprehensive, but alert. Paroxysmal cough, bringing up a little rusty streaked
sputum. Breathing with difficulty.
EENT exam?
Check all of the vital signs first!
Temperature?
103.
Blood pressure?
150/84.
Pulse?
100 a minute. Occasional extra systole.
...
Abdominal exam?
Diffusely tender, some splinting noticed in lower right quadrant and occasionally
radiating to right flank area.
The grippe.
There's insufficient information for any conclusive diagnosis at this time. You're
not being careful!
Musculo-skeletal system?
Essentially negative, except for generalized weakness and aching.
Extremities?
Normal, except nail beds are questionably cyanotic but difficult to determine because
of patient's skin color.
Rectal exam?
You're not being careful! The patient is too acutely ill for a rectal examination.
Laboratory reports?
Fine. Which lab tests would you like to have done?
RBC?
A red blood count will be done. Results will be sent to you as soon as possible.
...
```

Would you like to alert the night nurse to set up the lab tests you'll want to see tomorrow?
Yes.
Good. That will speed things up. However, there's nothing further that can be done tonight to help the patient. Let's call it quits for now. Perhaps you'll be in a better position to diagnose the case tomorrow after the first lab reports come in and you have reexamined the patient. Have a good night's sleep, and when you're back on duty tell me to "proceed."
Proceed.
It's morning and you're making your rounds. You've come to our patient and are given the night nurse's report: "Temperature remained high but patient spent a reasonably comfortable night. Cough diminished but persists. Occasional traces of blood in sputum. Patient still dyspneic and still has pain in abdomen. The second series of lab tests was initiated." Some of the first series of lab reports have just come in: urine exam, insufficient fluid for analysis; RBC, 5 million; WBC, 15500; differential, shift to left.
...
You may be prepared to make the diagnosis now. Are you?
No.
Fine. Nevertheless, we've got a fair amount of information—at least enough to raise some, perhaps several, diagnostic possibilities. Have you any in mind?
Yes.
Name one.
Pulmonary infarction.
Fine. Pleural pain, dyspnea, fever, and the physical exam signs and symptoms are certainly consistent with pulmonary infarction. Do you think that shaking chills and the presence of rusty sputum further support this diagnosis?
No.
Right. Though the white count is consistent, isn't it?
No.
Wrong. The white count is consistent. This should further support your diagnosis. We've considered one possibility. Do you have another diagnosis in mind?
No.
In that case I'd like to talk about viral pneumonia. The tachycardia, high WBC, elevated respiratory rate, shaking chills, bloody sputum, and severe pleural pain all lend weight to that diagnosis—right?
Yes.
You're way off base. There's not an iota of truth to any part of what I just said; in fact, the opposite of what I said is true. You'd better review some elementary medicine, doctor.
...
Pleural effusion.
A good contender, but let's be careful. What evidence would rule it in or out?
Positive evidence of another disease.
Is there such evidence?
Possible evidence though inconclusive.
Since you aren't sure, continue the examination until you have conclusive evidence one way or the other.
RBC?
An RBC is not indicated at this time. It's an unnecessary expense for a poor patient
WBC?
22500.

Staph pneumonia.
Fair enough. But where is the guilty organism? When you have isolated staphylococcus aureus in significant numbers in your samples of this patient's fluids or excretions, let me know.
Pneumococcal pneumonia.
Are you absolutely sure?
Yes.
You can't be, until you have irrefutable evidence. What would help you nail down the diagnosis?
Evidence from sputum smear and culture.
Good. Anything else?
No.
I think you might also want evidence from x-ray findings. Go ahead and we'll see what happens.
X-ray?
Not ready yet. Try the blood lab.
Differential?
Marked shift to left.
...
X-ray?
The x-ray shows: area of consolidation, right lower lobe, consistent with pneumatic process. This is a good time to review the total findings and to make another tentative diagnosis—if you like.
Evidence of fluid?
No evidence of fluid.
Sputum smear?
Gram-positive rods.
Strep pneumonia.
Fine, but how can you distinguish this from any other bacterial pneumonia? Wait until you have an irrefutable case.
Viral pneumonia.
But what about shaking chills, rusty sputum, elevated WBC, etc.—all the evidence of an acute febrile bacterial process? You are in error, doctor.
Sputum culture?
Presence of pneumococci.
Strep pneumonia.
Incredible! You have disregarded the positive findings that should clearly have established the correct diagnosis.
Pneumococcal pneumonia.
Very good. It's obvious now, though there were some confusing signs and symptoms in the picture. You've seen your way through to the correct diagnosis.

Bruce Jones programmed the computer to teach you about religion. Here's an example:

This program is designed to teach you some things about the so-called "documentary hypothesis" as formulated by Julius Wellhausen and others. First, what is your name?
George
All right, George, who wrote the first five books of the Bible, according to tradition?
Moses
You are off to a good start. Do you know what those first five books are called? They are called...
the Pentateuch
Right you are. Torah is the traditional Jewish name for them. They are also called the Pentateuch ("penta" = 5). People very early noticed that there was a problem with saying that Moses wrote the Torah. If you already know the basis of their suspicion, tell me.
Some things in the text reflect later conditions, after Moses.
You are right, if course, but there was an even more obvious problem. Try again.
I don't know.
At the end of the Pentateuch the author says, "Moses...died there." Does that seem strange to you?
yes
Why?
Moses could not have written about his own death.
O.k. Also, right after the report of Moses' death and burial (Deut. 34), the text says, "No man knows the place of his burial TO THIS DAY." How do those last three words strike you?
They seem later than the event they are talking about.
A good observation, George. Many people considered these words to be anachronistic. That is, "this day" is sharply distinguished from the time of the burial. It seems likely, therefore, that the time of writing was much later than the time of burial. Again (so the argument goes), Moses was not the author. Would these observations about the end of Deuteronomy convince you that Moses did not write the Torah?
yes
Why do you say that?
A man cannot write his own obituary.
Probably not many people would have been convinced if our evidence were so limited. After all, Moses could have written everything except the last chapter. However, as early as the 18th century, people noticed another problem....

The computer searched through the human's input, to see whether he used words indicating a correct answer. The computer never said a flat "Wrong", since religious answers are a matter of personal belief, and since the human might be smarter or weirder than the computer program was prepared for.

ROBOTS

In 1962 at MIT, Heinrich Ernst connected the computer to a mechanical hand that could feel. He made the hand build objects out of blocks, and made it put blocks into boxes.

* * *

One of the most famous robots is a guy named "Shakey", built at the Stanford Research Institute (SRI) in 1970. His eye contains a television camera (optical scanner). Instead of legs, he has wheels. Instead of arms, he has antennae (for feeling) and a bumper (for pushing). His brain is a computer: instead of carrying it around with him, he leaves it in another room and communicates with it by wireless methods.

To see how he works, suppose you type this message on his computer's terminal:
Push the block off the platform.
He begins by looking for the platform. If the platform is not in the room, he goes out to the hall, and steers himself through the hall (by looking at the baseboards) until he arrives at the next room. He peers into the room, to see whether it contains a platform. If not, he hunts for another room. When he finally finds a room containing a platform with a block on it, he tries to climb onto the platform to push the block off. But before climbing the platform, he checks the platform's height. If it is too high to get onto easily, he looks for a device to help him climb it. For example, if a ramp is lying in the room, he pushes the ramp next to the platform and then wheels himself up the ramp. Finally, he pushes the block off.

He can handle unexpected situations. For example, while he's getting the ramp, suppose you pull the platform to a different place. That doesn't faze him: he hunts for the platform again, and then pushes the ramp to it.

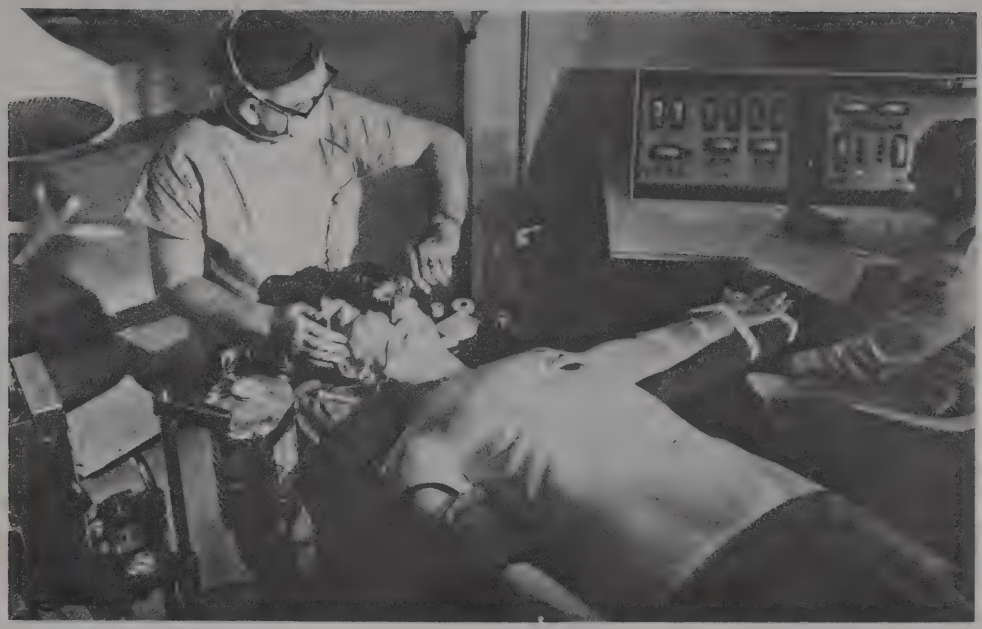
In 1971, Shakey's powers were extended, so he can handle commands such as:
Turn on the lightswitch.
If the lightswitch is too high for his bumper to reach, he looks for a device to climb onto, such as a box. If he finds a box that looks helpful, he climbs onto it, to check whether it is tall enough; if it is, he climbs off, pushes it to the lightswitch, climbs on it again, and finally flicks the switch.

Another task he can handle is:
Push three boxes together.
He finds the first box and pushes it to the second. Then he finds the third box, and pushes it to the second also.

He understands over 100 words. Whatever command you give him becomes his "goal", and he must reason out how to accomplish it. He might discover that to accomplish it, he must accomplish another goal first—for example, to move the block off the platform, he must first find the platform; to do that, he might have to look in another room; to do that, he must leave the room he's in; to do that, he must turn his wheels.

* * *

Here's a picture of a robot named Simulator One:



In the picture, a doctor is taking Simulator One's blood pressure and pulse. Another doctor is watching the computer console.

Simulator One is a model patient. He can blink, breathe, cough, vomit, respond to drugs, and even die. He is used in med school, to train doctors how to administer anaesthetics during surgery.

* * *

This report (abridged) is by Bertram Raphael, the director of the SRI Artificial Intelligence Center:

Here's what robots were capable of doing a few years ago.

At Hitachi Central Research Laboratory a TV camera was aimed at an engineering plan drawing of a structure. A second camera looked at blocks spread out on a table. The computer "understood" the drawing, reached towards the blocks with its arm, and built the structure.

At MIT, the camera was not shown a plan; instead, it was shown an example of the actual structure desired. The computer figured out how the structure could be constructed, and then built an exact copy.

At Stanford University, the hand obeyed spoken directions. For example, if someone said into the microphone, "Pick up the small block on the left," that is precisely what the arm would do.

In Scotland at the University of Edinburgh, a jumble of parts for two wooden toys was placed on a table. "Freddy," the Edinburgh robot, spread out the parts so that it could see each one clearly, and then, with the help of a vise-like work station at one corner of the table, assembled first the toy car and then the toy boat.

Recently, robot researchers have built robots that can perform truly practical tasks. For example:

At Stanford the system that used to stack toy blocks can now assemble a real water pump.

At SRI a computer-controlled arm with touch and force sensors can feel its way as it packs assembled pumps into a case.

At MIT programs are under development to enable a computer to inspect and repair circuit boards for use in computers, TV sets, and other electronic equipment.

* * *

Not all robots involve computers. Here's an example of a noncomputerized robot (reported by James Slagle, abridged):

A. George Carlton, John G. Chubbuck, and others at the Applied Physics Laboratory of Johns Hopkins University built a very interesting machine called The Beast. It is a battery-operated cylinder on wheels that is 18 in. in diameter. It has tactile, sonar, and optical apparatus. The sonar permits The Beast to find its way down the center of the hall. When its battery becomes sufficiently run-down, The Beast optically looks for an electric outlet and plugs itself in to recharge its battery. The Beast was often let loose to roam in the halls and offices at the Applied Physics Laboratory in order to see how long it could survive without "starving." Once it survived 40.6 hr. Many a new and unsuspecting secretary has been startled when The Beast entered her office, plugged itself into an electric outlet, and then departed. When it feels a step down, it knows enough to turn around, so that it does not fall downstairs. This logic however sometimes causes it to starve when it encounters a raised threshold. After getting on the threshold, it thinks it is about to fall, and so it turns around. After turning around it again thinks it is going to fall, and so it turns back and forth until it starves. It also starved when some workmen changed all the outlets from the flush to the projecting type. To cope with the new situation, the researchers changed some of the circuitry.

* * *

A recent newspaper article said that in Japan robots are being used in many practical ways. One robot arc-welds, reducing the time by 90%. Another grasps an object, determines the best way to pack it in a box, and does the packing; it uses television cameras and delicate arms. Another washes windows. Another wiggles a rod to catch a fish, takes the fish off the hook, dumps it into a bin, and returns the line to the water. Another directs traffic. Talking robots are being used instead of kimono-clad females in inns and restaurants.

Commenting on the quality of life in Japan, the article went on to say that people are buying whiffs of oxygen from vending machines.

The article was tacked on the bulletin board at the MIT Artificial Intelligence Laboratory, together with this graffito about how the Japanese robots would act differently if they were as smart as people....

HUMAN: Weld these parts.
ROBOT: The steel in those parts is grossly inferior. They must have been made in the U.S. Send them back. Also, have my welding machine tuned up.
HUMAN: Pack those widgets.
ROBOT: Can I break them in half first?
HUMAN: No.
ROBOT: "No" is not an allowed answer. I only have small shipping boxes. (Proceeds to break widgets in half and stuff them into boxes.)
HUMAN: Wash those windows.
ROBOT: What?? And get my precious electrical and mechanical components all wet??
HUMAN: Catch a fish.
ROBOT: (Proceeds to catch a fish, take it off the hook, and throw it back.) Okay.
HUMAN: What did you throw it back for?
ROBOT: It was under the size limit. Anyway, it was full of mercury.
HUMAN: Direct that traffic.
ROBOT: Will I get hazardous duty pay for this work?
HUMAN: Greet those customers.
ROBOT: Sorry, my shift is over.
HUMAN: (Ready to faint.) Give me some air.
ROBOT: Use that "machine" over there. (Points to vending machine.)

CREATE INTELLECT

CATEGORIES

The field of "artificial intelligence" consists of several categories. Let's examine them.

MENTAL GAMES

Can the computer play chess and checkers better than humans? That question has a long history....

1950: Shannon invents the "minimax tree method", to help the computer win
1962: Samuel's checker program wins a game of checkers against Robert Nealey, an expert
1967: Greenblatt writes the first decent program for chess
1976: chess program by Atkins & Gorklen does well in championships against humans
1976: Jennings writes Microchess, a program that consumes only 1K of RAM
1978: the Sprachlins write Sargon, the first microcomputer chess program that plays well

NATURAL LANGUAGES

Can the computer understand English, Russian, and the other languages that humans speak? Such languages are called *natural languages*, as opposed to the restrictive *computer languages* in which programs are written.

Here are the highlights....

1954: IBM writes a program that translates simple sentences from Russian to English
1961: MIT makes the computer answer questions about baseball, asked in ordinary English
1964: Bobrow makes the computer solve story problems about arithmetic and algebra
1964: SDC makes the computer understand a child's encyclopedia
1966: Weizenbaum's "Eliza" program makes the computer act as a psychotherapist
1971: Colby makes the computer act paranoid

INTERPERSONAL RELATIONSHIPS

The computer can imitate interpersonal relationships. Here's a report by Nigel Hawkes (abridged):

John Loehlin at the University of Texas devised a program that imitates human personality. In honor of the author of Brave New World, he calls the model Aldous. It contains five subroutines, concerned with recognition, emotional reaction, action selection, introspection, and learning, and two memories, a small immediate one and a larger long-term one. When presented with an input, Aldous responds to it "emotionally" with fear, anger, or attraction, and generates actions of withdrawal, approach, attack, conflict, or indifference. Aldous's personality can be altered by modifying the subroutines. Thus Aldous programmed one way will respond warmly to emotion ("romantic Aldous") or to abstract or concrete aspects of a situation ("intellectual Aldous") or alternatively be dominated by the past ("conservative Aldous").

Objects presented to Aldous must be described by a simple numerical code, so that Aldous does not require complex recognition equipment. When he's presented with the same object four times in a row, he gets attached to it and his attitude is positive. When presented with a new object which is very similar, his positive emotions carry over, and he approaches it with confidence; but if it doesn't have the same favorable consequences to him, he learns eventually to be hostile to it.

Loehlin programmed two different Aldous models into the computer at once and made them interact with each other. The social relationship turned out to be rather delicate. If either model started out in a negative frame of mind, the relationship tended to turn rapidly into mutual hostility. If both models were favorably disposed, a harmonious relationship might develop, but it was far from certain. It tended to go through phases—a warm period to start with (the honeymoon phase) followed by a phase in which each model had to cope with hostility from the other, and finally a stable relationship with increasingly positive attitudes from both models.

In another experiment, Loehlin had one Aldous interact with another programmed to produce unshakeably positive responses ("Saint Aldous"). Violent aggression resulted.

In an experiment where one model played the part of a psychotherapist, the patient, initially strongly negative, was eventually brought round to a positive attitude, at least towards the therapist. Replacing the therapist by an ordinary Aldous was not easy, however, and another period of therapy was needed before a stable and positive relationship could be established.

In a model called Homunculus, produced by John and Jeanne Gullahorn of Michigan State University, three-person interactions are possible. The model uses a theory of human relations devised by George Homans. When three unacquainted models are "introduced", they typically form a pair and an isolate; when the models have prior knowledge about each other, more complex interactions may occur.

THE ARTS

In 1956 at the University of Illinois, Lejaren Hiller and Leonard Isaacson made the computer compose its own music. The computer picked only "white" notes— no sharps or flats. It picked each note by using random numbers; but before printing the note, the computer checked whether that note would contradict any of the fourteen rules of counterpoint; if so, the computer picked a different note instead. Hiller says the music is "of fair quality, strongly reminiscent, if one ignores a certain monotony in rhythm, of passages from Palestrina."

In 1957 they tried another experiment: they allowed sharps and flats, and fed in only four of the fourteen rules. The result was chromatic music. According to Hiller, it "recalled passages from, say, a Bartok string quartet." They also made the computer compose twelve-tone music in the style of Schoenberg.

Can the computer compose its own poetry and prose? Here are the computer's accomplishments....

1961: Balestrini's program writes "The Taoist Hiroshima Elevator", by combining sources
1966: Northwestern University makes the computer compose a poem about bubbles
1968: Mendoza makes the computer write children's stories and fake physics essays
1968: Deffrey makes the computer write fake essays about the aerospace industry
1972: Gosden makes the computer produce "Me-Books"— personalized stories for kids
1973: I make the computer compose stories— some are pornographic
1973: D'Oench makes the computer compose poetic images
1978: Dwyer & Critchfield make the computer curse you

The computer has also drawn original artwork, by using random numbers and other techniques.

PROTOCOL METHOD

During the 1950's and 1960's, most research in artificial intelligence was done at the Massachusetts Institute of Technology (MIT) and the Carnegie Institute of Technology (CIT, now called Carnegie-Mellon University). At Carnegie, the big names were Allen Newell and Herbert Simon. They invented the *protocol method*. In the protocol method, a human is told to solve a tough problem and, while he's solving it, to say at each moment what he's thinking. A transcript of his train of thought is recorded and called the *protocol*. Then programmers try to make the computer imitate that train of thought.

Using the protocol method, Newell and Simon produced programs that could "think like humans". The thinking, like human thinking, was imperfect. Their research did *not* try to make the computer a perfect thinker; instead, it tried to gain insight into how *humans* think. Their point of view was: if you think you really understand human psychology, go try to program it. Their attempt to reduce human psychology to computer programs is called *mentalism*, and has replaced Skinner's stimulus-response behaviorism as the dominant force in psychology today.

ABSTRACT MATH

Many programmers have tried to make the computer do abstract math.

In 1957 Newell, Simon, and Shaw used the protocol method to make the computer prove theorems about symbolic logic, such as "Not (p or q) implies not p". In 1959 and 1960, Herbert Gelernter and his friends made the computer prove theorems about Euclidean geometry, such as "If the segment joining the midpoints of the diagonals of a trapezoid is extended to intersect a side of the trapezoid, it bisects that side." In 1961, MIT awarded a Ph.D. to James Slagle for making the computer compute indefinite integrals, such as $\int \frac{x^4}{(1-x^2)^{5/2}}\,dx$; the computer gets the answer, which is $\arcsin x + \frac{\tan^3 \arcsin x}{3} - \tan \arcsin x + c$.

Each of those programs works by drawing a tree inside the computer's memory. Each branch of the tree represents a possible line of attack. The computer considers each branch and chooses the one that looks most promising.

A better symbolic-logic program was written by Hao Wang in 1960. His program doesn't need trees; it always picks the right attack immediately. It's guaranteed to prove any theorem you hand it, whereas the program by Newell, Simon, and Shaw got stuck on some hard ones.

A better indefinite integration program was written by Joel Moses in 1967 and further improved in 1969. It uses trees very rarely, and solves almost any integration problem.

A program that usually finds the right answer but might fail on hard problems is called *heuristic*. A heuristic program usually involves trees. The checkers, chess, and geometry programs are heuristic. A program that's guaranteed to always give the correct answer is called *algorithmic*. The original symbolic-logic program was heuristic, but Wang's improvement is algorithmic; Moses's indefinite integration program is almost algorithmic.

GPS

In 1957 Newell, Simon, and Shaw began writing a single program to solve *all* problems. They called the program *GPS (General Problem Solver)*. If you feed the program a goal, a list of operators, and associated information, the program will tell you how to achieve the goal by using the operators.

For example, suppose you want the computer to solve this simple problem: a monkey would like to eat some bananas that are too high for him to reach, but there's a box nearby he can stand on. How can he get the bananas?

Feed the GPS program this information....

Now: monkey's place = $place_1$; box's place = $place_2$; contents of monkey's hand = empty
Want: contents of monkey's hand = the bananas
Difficulties: contents of monkey's hand is harder to change than box's place, which is harder to change than monkey's place

Allowable operator	Definition
climb box	before: monkey's place = box's place after: monkey's place = on the box
walk to x	after: monkey's place = x
move box to x	before: monkey's place = box's place after: monkey's place = x; box's place = x
get bananas	before: box's place = under the bananas; monkey's place = on the box after: contents of monkey's hand = the bananas

GPS will print the solution:

walk to $place_2$
move box to under the bananas
climb box
get bananas

The GPS approach to solving problems is called *means-ends analysis*: you tell the program the means (operators) and the end (goal). The program has proved theorems in symbolic logic, computed indefinite integrals, and solved many famous puzzles, such as "The Missionaries and the Cannibals", "The Tower of Hanoi", and "The 5-Gallon Jug and the 8-Gallon Jug". But the program works slowly, and you must feed it lots of information about the problem. The project was abandoned in 1967.

VISION

Another large topic in artificial intelligence is *computer vision*: making the computer see.

The first problem tackled was *pattern recognition*: making the computer read handwritten printed letters. The problem is hard, because some people make their letters very tall or wide or slanted or curled or close together, and the pen may skip. Reasonably successful programs were written, although computers still can't tackle script.

Interest later shifted to *picture processing*: given a photograph of an object, make the computer tell what the object is. The problem is hard, because the photo may be taken from an unusual angle and be blurred, and because the computer gets confused by shadows.

Scene analysis is even harder: given a picture of a group of objects, make the computer tell which object is which. The problem is hard, because some of the objects may be partly hidden behind others, and because a line can have two different interpretations: it can be a crease in one object, or a dividing-line between two objects.

Most of the research in picture processing and scene analysis was done rather recently: from 1968 to 1972.

Raymond Kurzweil has invented an amazing machine whose camera looks at a book and reads the book, by using a voice synthesizer. Many blind people use it.

ROBOTS

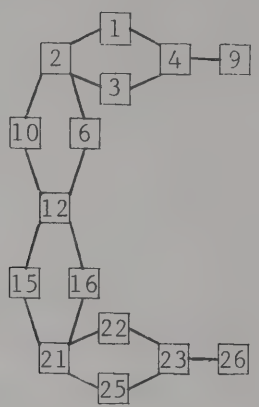
Researchers have built robots. The first robots were just for experimental fun, but today's robots are truly useful: for example, the Japanese are using robots to manufacture cars. In the United States, many young kids are being taught "LOGO", which is a language developed at the MIT Artificial Intelligence Laboratory that makes the computer control a robot turtle.

TODAY'S RESEARCH

Today, research in artificial intelligence is done at four major universities: MIT, Carnegie, Stanford, and Edinburgh (Scotland).

REFLEXIVE CONTROL

In the Soviet Union, weird researchers have studied *reflexive control*: they've programmed the computer to be disobedient. The first such programmer was Lefevr, in 1967. In 1969 Baranov and Trudolyubov extended his work, by making the computer win this disobedience game:



The human begins by choosing either node 9 or node 26, *but doesn't tell the computer which node he's chosen*. The computer starts at node 12; on each turn, it moves to an adjacent node. When it reaches either node 9 or node 26, the game ends: if the node the computer reaches is the one the human chose, the human wins; if the computer reaches the opposite node, the computer wins. Before each move, the human tells the computer where to go; but the computer may decide to do the opposite (disobey).

What strategy should the computer use? If it always obeys, or always disobeys, the human will catch on, and will make it lose.

Instead, Baranov and Trudolyubov programmed the computer to react as follows:

obey the human twice, then disobey three times, then obey once, disobey thrice, obey once, disobey twice, obey thrice, disobey once, obey thrice, disobey once,...

The irregular alternation of obedience and disobedience confuses the human in a way that works to the computer's advantage. Using that strategy, the computer played against 61 humans, and won against 44 of them (72%). In other words, the typical human tried to mislead the computer but in fact "clued it in" to the human's goal.

Later experiments with other games indicated that the following pattern of disobedience is usually more effective:

obey the human twice, disobey thrice, obey once, disobey four times, obey once, disobey thrice, obey thrice, disobey twice, obey thrice, disobey once, obey once, disobey once

MISINFORMATION

The top researchers in artificial intelligence often exaggerate their achievements, and underestimate how long it will take to develop a truly intelligent computer. Don't believe everything they say.

In 1957 Herbert Simon said, "Within ten years a digital computer will be the world's chess champion." In 1967, when the ten years had elapsed, the only decent chess program was Greenblatt's, which the American Chess Federation rated "class D" (which means "poor"). Today, the best chess program is ranked as an "expert" or "master"; it's far less than a "senior master" or "international master" or "grandmaster" or "world champion".

In 1957 Simon also said, "Within ten years a digital computer will discover and prove an important new mathematical theorem." He was wrong. The computer still hasn't discovered or proved any important new mathematical theorem. The closest call came in 1976, when it did the *non-abstract part* of the proof of the "4-color theorem".

In 1958 Newell, Simon, and Shaw wrote chess-playing program which they admitted was "not fully debugged" so that one "cannot say very much about the behavior of the program"; but they claimed it was "good in spots (opening)". In 1959 the founder of cybernetics, Norbert Weiner, exaggerated about their program; he told New York University's Institute of Philosophy that "chess-playing machines as of now will counter the moves of a master player with the moves recognized as right in the textbooks, up to some point in the middle game." In the same symposium Michael Scriven carried the exaggeration even further by saying, "Machines are already capable of a good game." In fact, the program they were describing played very poorly, and in its last official bout (October 1960) was beaten by a ten-year-old kid who was a novice.

In 1960 Herbert Gelernter (who wrote the geometry-theorem program) said, "Today hardly an expert will contest the assertion that machines will be proving interesting theorems in number theory three years hence." More than twenty years have elapsed since then, but neither Gelernter nor anyone else has programmed the computer to prove theorems in number theory.

In June 1963 this article appeared in the *Chicago Tribune*:

The development of a machine that can listen to any conversation and type out the remarks just like an office secretary was announced yesterday by a Cornell University expert on learning machines. The device is expected to be in operation by fall. Frank Rosenblatt, director of Cornell's cognitive systems research, said the machine will be the largest "thinking" device built to date. Rosenblatt made his announcement at a meeting on learning machines at Northwestern University's Technological Institute.

No such machine exists today, let alone in 1963.

Also in 1963, W. Ross Ashby said, "Gelernter's theorem-proving program has discovered a new proof of the *pons asinorum* that demands no construction." He said the proof is one that "the greatest mathematicians of 2000 years have failed to notice... which would have evoked the highest praise had it occurred." In fact, the *pons asinorum* is just the simple theorem that the opposite angles of an isosceles triangle are equal, and the computer's constructionless proof had already been discovered by Pappus in 300 A.D.

In 1968 the head of artificial intelligence in Great Britain, Donald Michie, said, "Today machines can play chess at championship level." In fact, when computers were allowed to participate in human chess tournaments, they almost always lost.

In 1970 the head of artificial intelligence at MIT, Marvin Minsky, said:

In from three to eight years we will have a machine with the general intelligence of an average human being. I mean a machine that will be able to read Shakespeare, grease a car, play office politics, tell a joke, have a fight. At that point, the machine will begin to educate itself with fantastic speed. In a few months it will be at genius level, and a few months after that its powers will be incalculable.

His prediction that it would happen in three to eight years— between 1973 and 1978— was ridiculous. I doubt it will happen during this century, if ever.

Exaggerations concern not only the present and future but also the past. Back in 1962 Arthur Samuel's checker program won one game against Robert Nealey, "a former Connecticut checkers champion". Notice that Nealey was a *former* champion, not *the current* champion when the game was played. Also notice the program won a single game, not a match; and in fact it lost to Nealey later. In 1971 James Slagle slid over those niceties, when he just said that the program "once beat the champion of Connecticut." More recent writers, reading Slagle's words, have gone a step further and omitted the word *once*: one textbook says, "The current program beat the champion of Connecticut". It's not true.

Why do leaders of artificial intelligence consistently exaggerate? The answer is obvious: to get more research funds from the government. Hubert Dreyfus, chairman of the philosophy department at Berkeley, annoys them by attacking their claims.

THE BRAIN

Will the computer be able to imitate the human brain? Opinions vary.

Marvin Minsky, head of artificial intelligence at MIT, says *yes*: "After all, the human brain is just a computer that happens to be made out of meat."

Biologists argue *no*: the brain is composed of 12 billion *neurons*, each of which has between 5,000 and 60,000 *dendrites* for input and a similar number of *axons* for output; the neurons act in peculiar ways, and no computer could imitate all that with complete accuracy— "The neuron is qualitatively quite different from 'on-off' components of current computers."

Herbert Simon (head of artificial intelligence at Carnegie and a psychologist), points out that certain aspects of the brain, such as short-term memory, are known to have very limited capacity and ability. He believes the inner workings of the brain are reasonably simple; it produces complicated output only because it receives complicated input from the sense organs and environment: "A man, viewed as a behaving system, is quite simple. The apparent complexity of his behavior over time is largely a reflection of the complexity of the environment in which he finds himself." Simon believes that if a computer were given good sense organs, the ability to move, and an elementary ability to learn, and were placed in a stimulating environment (unlike the dull four walls of a computer center), it would start acting in complex ways also.

Hubert Dreyfus, chairman of the philosophy department at Berkeley, argues that progress in artificial intelligence has been very small, is being blocked now by impenetrable barriers, and— most important— the computer's approach to solving problems bears little relationship to the more powerful methods used by humans. He's cynical about the claim that an improvement in computer programs represents progress toward understanding the human mind, which is altogether different: "According to this definition, the first man to climb a tree could claim tangible progress toward reaching the moon. Rather than climbing blindly, it's better to look where one is going."

APPENDIX I

PECULIARITIES

APPLE

SCOPE

This section explains Apple's new, popular computers: the Apple 2-plus and the Apple 2e. (If you have an old Apple 2 instead, make it act like an Apple 2-plus, by inserting an Applesoft ROM and an auto-start ROM.)

THE KEYBOARD

On the Apple 2-plus, the zero key is Swedish; but on the Apple 2e, the zero key is *not* Swedish.

The BACKSPACE key has a left-arrow on it. To correct an error, press the left-arrow key, until you get back to where the error was. Then correct the error, by retyping. Then, if you wish, press the right-arrow key to move to the right.

The Apple 2e also has a key marked DELETE, but the DELETE key doesn't work. To backspace, use the left-arrow key, not the DELETE key.

GET STARTED

To start using the Apple, you can choose either the "easy" way or the "fancy" way. The "easy" way is easier to understand; the "fancy" way lets your Apple do extra tricks.

If you choose the "easy" way, do *not* use a disk: if your Apple is attached to disk drives, remove any disks from the drives.

If you choose the "fancy" way, you need a disk drive, and you must make sure it contains a special kind of disk. The disk should be a "DOS 3.3 system master", or a *copy* of a DOS 3.3 system master, or a *slave* of a DOS 3.3 system master. Put the disk into the *main* disk drive. (If you have two disk drives, the main drive should be to the left of the other drive.) When you put the disk into the drive, make sure the disk's label is on *top* of the disk, and make sure the disk's big oval cutout goes into the drive *before* the label does. After putting the disk into the drive, close the drive's door.

Attach the computer to a TV or to a monitor. Turn on the TV or monitor. (If you're using a TV, turn it to channel 33.) Then flip the computer's power switch to the ON position. (The power switch is on the back of the computer.)

If you have a disk drive, the drive's red light will turn on and the drive will make noise, but ignore the light and the noises.

On the screen of the TV or monitor, you should see some writing. If you're using a TV and the writing is fuzzy, try switching channels: instead of channel 33, try 32 or 34 or 10, and adjust the fine tuning.

Adjust the screen's contrast and brightness.

If you used the "easy" way (which lacks disks), do a "controlled reset". Here's how. . . . Hold down the CONTROL key. (The Apple 2e's CONTROL key says "CONTROL" on it. The Apple 2-plus's CONTROL key says "CTRL" on it.) While you keep holding down the CONTROL key, tap the RESET key. The computer will beep.

When the computer is on and functioning correctly, you'll see this symbol on the screen, near the screen's *left* side:

]

That symbol is called a *bracket*. It means everything's OK. (The Apple prints a bracket instead of the word OK.)

Finally, if you have an Apple 2e, press the CAPS LOCK key, to make sure it's in the DOWN position.

PROGRAMMING

Make sure you type the word NEW above your program. If you forget to type the word NEW, and it's your first program, and you've been using disks, the computer will combine your program with the computer's "disk start-up program", and you'll have a mess!

HOW TO PRINT ON PAPER

Make sure the printer is plugged in, turned on, full of paper, and ready on-line.

To activate the printer, type this:

```
PR#1
```

That makes the computer use paper instead of your screen. Henceforth, the computer will print all its answers on paper instead of on your screen.

If you ever want the computer to print on your screen again, you must deactivate the printer, by typing:

```
PR#0
```

For example, to make the computer print just "I LOVE YOU" on paper, you must activate the printer, then tell the computer to print "I LOVE YOU", then deactivate the printer, like this:

```
PR#1
PRINT "I LOVE YOU"
PR#0
```

If a program prints on the screen but you want it to print on paper instead, activate the printer before you type RUN. Here's how:

```
PR#1
RUN
PR#0
```

Here's a fancy program that prints *two* poems. It prints "LOVE IS LIKE A DOVE" on your screen, and prints "LIFE IS FULL OF STRIFE" on paper:

```
NEW
1 PRINT "LOVE"
2 PRINT "IS LIKE A DOVE"
3 PR#1
4 PRINT "LIFE"
5 PRINT "IS FULL OF STRIFE"
6 PR#0
RUN
```

REARRANGING YOUR PROGRAM

To LIST your program on paper instead of on your screen, type this:

```
PR#1
LIST
PR#0
```

Instead of DELETE, you must say DEL. To delete lines 30 through 80, type this:

```
DEL 30,80
```

SPACES

The Apple does *not* automatically put spaces near numbers.

For example, if you give this command—

```
PRINT 6+2;6-2
```

the Apple will print the answer to 6+2 (which is 8) and also print the answer to 6-2 (which is 4), but won't put any spaces between the answers; the Apple will print:

```
84
```

To force the Apple to put a space between the answers, replace the semicolon by a *space in quotation marks*, like this:

```
PRINT 6+2" "6-2
```

That makes the Apple print:

```
8 4
```

CLEARING THE SCREEN

The Apple lacks a CLEAR key. To clear the screen, press the ESC key, then type an @ symbol. (To type the @ symbol, remember to press the SHIFT key.)

The Apple doesn't understand the CLS command. Instead of typing CLS, type the word HOME.

GO TO

The BREAK key is marked RESET; but it works only if you simultaneously hold down the CONTROL key (which is marked CONTROL or CTRL). So to abort your program, hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the RESET key.

If the computer is printing on the screen faster than you can read, you can make the computer slow down. To slow the printing, type this command:

```
SPEED=0
```

For example, to make the computer LIST your program slowly, and also PRINT slowly when it runs your program, type this:

```
SPEED=0
LIST
RUN
```

To make the computer print quickly again, at full speed, type this command:

```
SPEED=255
```

SPEED=0 means slow; SPEED=255 means fast. For an intermediate speed, pick a number between 0 and 255.

NUMERIC VARIABLES

A numeric variable can be a letter, or a *pair* of letters, or a letter-followed-by-a-digit. For example, in this program the variable is a pair of letters (MA):

```
10 MA=8
20 PRINT MA+1
```

That program makes the computer print 9.

Any pair of letters is okay, except the following pairs (which have special meanings): TO, IF, ON, FN, GR, AT, OR.

STRING VARIABLES

A string variable looks like a numeric variable, but has a dollar sign after it. For example, you can have a string variable such as MA$.

STRING INPUT

In the INPUT statement, the computer's question is called the *prompt*. The prompt begins and ends with a quotation mark. On the Apple, the prompt should end with *a question mark then a space then a quotation mark*. . . .

Most computers:
```
10 INPUT "WHAT IS YOUR NAME";N$
```
Apple:
```
10 INPUT "WHAT IS YOUR NAME? ";N$
```

FANCY CONDITIONS

The Apple can't handle the letter A followed by the word THEN. For example, it can't handle:

```
IF X=A THEN PRINT "WOW"
```

To prevent the letter A from being next to THEN, reverse the equality—

```
IF A=X THEN PRINT "WOW"
```

or use parentheses to separate the A from the THEN:

```
IF (X=A) THEN PRINT "WOW"
```

DISKS

How to prepare the Apple for handling disks

At the beginning of this Apple appendix, you'll find a section entitled "GET STARTED". It explains how to get started using the Apple. It says there are two ways: an easy way, and a fancy way.

To use disks, you must start the Apple by using the fancy way. For details, reread the section entitled "GET STARTED".

If you started the Apple by using the easy way instead, turn the Apple off and then restart the Apple by using the fancy way.

How to turn a blank disk into an Apple disk

If you buy a blank disk and want the Apple to use it, you must turn the blank disk into an Apple disk.

To do that, insert the blank disk into the main disk drive (after removing whatever disk was in there before). Then type the following:

```
NEW
10 PRINT "HELLO"
INIT HELLO
```

The disk drive's red light will glow for about 30 seconds. When it stops glowing, your blank disk is finally an Apple disk; it's called a "slave of a DOS 3.3 system master".

Apple disk commands

For the Apple, omit quotation marks:

Most computers	Apple
SAVE "JOE"	SAVE JOE
LOAD "JOE"	LOAD JOE
RUN "JOE"	RUN JOE

Apple's vocabulary differs from other computers:

Most computers	Apple
KILL "JOE"	DELETE JOE
FILES	CATALOG

Your program's name can be short (such as JOE) or long: up to 30 characters. The name's first character must be a letter; the remaining characters can be letters, digits, spaces, math symbols, punctuation marks, or any other keystrokes you wish—except that you must not include any commas.

How to select a drive

The main disk drive (which is on the left) is called *drive 1*; the other disk drive (which is on the right) is called *drive 2*. Normally, the Apple uses just drive 1.

If you want it to use drive 2 instead, put a comma and a D2 at the end of the disk command. For example, to get a CATALOG of drive 2, say:

```
CATALOG,D2
```

To save JOE in drive 2, say:

```
SAVE JOE,D2
```

If you put a comma and D2 at the end of a command, the Apple assumes you're more interested in drive 2 than in drive 1. Therefore, it will automatically use drive 2 instead of drive 1 for all future commands. If you ever want to use drive 1 again, you must give a command that ends in a comma and D1.

How to copy an entire disk

To copy an entire disk, put a DOS 3.3 system master disk into the main drive. Type RUN COPYA and press the RETURN key. The disk drive's red light will glow. When the light stops glowing, remove the DOS 3.3 system master disk from the main drive.

Into the main drive, put the disk you want to copy. Into the other drive, put any blank disk. (It doesn't have to be an "Apple" disk.)

Press the RETURN key five times. The computer will copy all information from the main drive's disk to the blank disk. (In the process, it will also turn the blank disk into an "Apple" disk.)

Then the Apple will ask:

```
DO YOU WISH TO MAKE ANOTHER COPY?
```

If you don't, type an N and then press the RETURN key.

TAPES

How to copy a program onto a tape

After positioning the tape and noting the counter's number, do the following . . .

Type the word SAVE, without pressing the RETURN key. On the recorder, press the RECORD and PLAY buttons, at the same time as each other. Then press the computer's RETURN key.

The computer will start copying the program onto tape. The computer will give a beep.

When the computer finishes copying, it will beep again. Then it will print the symbol "]". Press the recorder's STOP button.

How to read a program from a tape

Put the tape into the recorder. (To do that, you might have to press the recorder's EJECT button.)

Press the recorder's REWIND button. When the tape is rewound, press the recorder's STOP button.

Push the counter's button, so the counter becomes zero.

Press the FAST FORWARD button. Look at the counter: when the number on the counter is almost up to the number of the program's beginning, press the STOP button.

Type the word LOAD, without pressing the RETURN key. Press the PLAY button. When the counter reaches the number of your program's beginning, press the RETURN key.

The computer will start reading the tape. The computer will give a beep.

When the computer finishes reading the program, it will beep again and then print the symbol "]". Press the STOP button.

Type the word LIST or RUN, then press the RETURN key.

EDITING

If you make a mistake while typing a line, you can use the left-arrow and right-arrow keys to correct the mistake. (In addition to the left-arrow and right-arrow keys, the Apple 2e also has a DELETE key; but the DELETE key doesn't work!)

The Apple 2-plus has neither an up-arrow key nor a down-arrow key. The Apple 2e has both an up-arrow key and a down-arrow key, but those keys don't work! Therefore, correcting an error in a previous line is difficult. Here's how. . . .

To edit line 20, type:

```
LIST 20
```

Press the ESCape key. Press the J key. Press the I key repeated, until you get up to the beginning of line 20. Press the right-arrow key. Press the right-arrow key several more times, until you get to where the error is. Retype that part of the line correctly; if you want that part of the line to be just blank, hit the SPACE bar repeatedly. Press the right-arrow key repeatedly, until you get *past* the end of line 20. Press the RETURN key twice.

RENUMBERING

Instead of typing the word RENUM, type an ampersand (which is the symbol "&"). After typing the ampersand, press the RETURN key.

But before typing the ampersand, you must teach the computer the *meaning* of the ampersand. To do that, SAVE whatever program you're working on; then put a DOS 3.3 system master disk into the main drive, type RUN RENUMBER, and press the RETURN key twice.

Once you've taught the computer the definition of the ampersand, the computer will remember the definition until you turn off the computer's power.

PAUSING

To make the Apple pause, hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the S key.

To stop pausing, tap any key.

APOSTROPHE

Instead of an apostrophe, type a colon followed by the word REM (which stands for "remark"). For example:

```
40 C=30: REM BECAUSE RUSS HAS 30 COMPUTERS
```

COUNTING TO 100

If you indent the lines between FOR and NEXT, the Apple ignores the indentation. When you LIST your program, you won't see any indentation. So don't bother to indent.

ZONES

The Apple has only 3 zones. The left zone is 16 characters wide; the middle zone is also 16 characters wide; the right zone is only 8 characters wide. So the entire screen is 40 characters wide.

LOCATE

The Apple doesn't understand the word LOCATE. To print "DROWN" beginning at line 3's 7th position, say this:

```
VTAB 3: HTAB 7: PRINT "DROWN"
```

RANDOM NUMBERS

For a random integer between 1 and 5, say 1+INT(RND(1)*5) instead of RND(5). For a random decimal between 0 and 1, say RND(1). Omit the word RANDOMIZE; the Apple doesn't understand it.

Low-resolution graphics

To use low-resolution graphics, type:

```
GR
```

That makes the computer split the screen into two parts. The top part of your screen will be used for low-resolution graphics. The bottom of your screen will *not* be used for graphics: instead, it will be used for non-graphics (which Apple calls "text").

When you type GR, the computer makes the top part of your screen (the graphics part) be entirely black.

Your next step is to choose a color to draw in. You can choose these colors:

0. black
1. magenta (purplish red)
2. dark blue
3. lavender (light purple)
4. dark green
5. gray
6. blue
7. light blue
8. brown
9. orange
10. gray
11. pink
12. green
13. yellow
14. aqua (light greenish blue)
15. white

You must also choose which pixel you want to color: the pixel's X and Y coordinates must each be from 0 to 39. For example, to make pixel 31,37 become pink, type this:

```
COLOR=11: PLOT 31,37
```

To make pixel 5,9 and pixel 17,21 both become yellow, type this:

```
COLOR=13: PLOT 5,9: PLOT 17,21
```

If the colors on your TV look wrong, adjust your TV's COLOR and TINT dials.

To draw a green horizontal line from 4,19 to 8,19, type this:

```
COLOR=12: HLIN 4,8 AT 19
```

To draw a blue vertical line from 4,20 to 4,30, type this:

```
COLOR=6: VLIN 20,30 AT 4
```

To print "WELCOME TO THE ART MUSEUM" on the bottom part of the screen (which is for non-graphics), just say:

```
PRINT "WELCOME TO THE ART MUSEUM"
```

Whenever you want to stop using graphics, type:

```
TEXT: HOME
```

That tells the computer to use the entire screen for non-graphics (TEXT).

You can put all those ideas into a program, like this:

```
10 HOME: GR
20 COLOR=11: PLOT 31,37
30 COLOR=13: PLOT 5,9: PLOT 17,21
40 COLOR=12: HLIN 4,8 AT 19
50 COLOR=6: VLIN 20,30 AT 4
60 PRINT "WELCOME TO THE ART MUSEUM"
70 FOR I = 1 TO 4000: NEXT
80 TEXT: HOME
```

Line 10 includes the word HOME, to make sure that the bottom part of the screen (for non-graphics) begins entirely black. Lines 20-50 draw a picture. Line 60 prints a caption. Line 70 makes the computer count to 4000: while the computer counts, you can admire the picture that the computer drew; if you want to admire the picture longer, change the 4000 to 10000. Line 80 returns the computer to its normal state, which is non-graphics.

In that program, lines 10, 70, and 80 are particularly important. Every low-resolution graphics program should begin by saying "HOME: GR", and should end by saying "FOR I" and "TEXT: HOME".

High-resolution graphics

High-resolution graphics differs from low-resolution graphics in the following ways. . . .

For low-res, you say "GR". For high-res, you say "HGR: VTAB 21".

For low-res, you say "COLOR=" and choose a color from 0 to 15. For high-res, you say "HCOLOR=" and choose one of these high-res colors:

0. *black*
1. *green*
2. *violet*
3. *usually blue, if X coordinate even; usually green, if X coordinate odd; white, if adjacent pixel plotted also*
4. *black*
5. *orange*
6. *blue*
7. *white*

For low-res, you say "PLOT". For high-res, you say "HPLOT".

For low-res, the X coordinate and Y coordinate each go from 0 to 39. For high-res, the X coordinate goes from 0 to 279, and the Y coordinate goes from 0 to 159.

For low-res, you say "HLIN" to draw a horizontal line, and you say "VLIN" to draw a vertical line. For high-res, you can draw a line in *any* direction, by saying "HPLOT TO". For example, if you say—

```
HPLOT 250,140
HPLOT TO 70,90
HPLOT TO 180,114
```

the computer will plot the pixel 250,14, then draw a diagonal line from there to pixel 70,90, then draw a diagonal line from pixel 70,90 to pixel 180,114. You can combine those commands into a single command:

```
HPLOT 250,140 TO 70,90 TO 180,114
```

Here's a complete program:

```
Use high-res.          10 HOME: HGR: VTAB 21
Draw green lines.      20 HCOLOR=1: HPLOT 250,140 TO 70,90 TO 180,114
Write a caption.       30 PRINT "WELCOME TO THE ART MUSEUM"
Pause.                 40 FOR I = 1 TO 4000: NEXT
End the graphics.      50 TEXT: HOME
```

MUSIC

To make the Apple play music, you need to use a different computer language: BASIC isn't fast enough. Most Apple music programs are written in assembly language instead of in BASIC.

EXPONENTS

For exponents, Apple uses the symbol "^" instead of "↑".

Unfortunately, the Apple makes one kind of mathematical error when handling exponents. According to mathematicians and most other computers, -5^2 means "the negative of 5^2" and is therefore -25. But the Apple mistakenly thinks that -5^2 means "the square of -5", and therefore erroneously prints 25.

For another example, suppose you say:

```
10 X=5
20 PRINT -X^2
```

The Apple mistakenly thinks that line 20 means "the square of -X", and therefore prints 25. (According to mathematicians and other computers, line 20 ought to mean "the negative of X^2", and so the answer ought to be -25.)

The Apple's mistake will make the Apple incorrectly handle several branches of mathematics. For example, the Apple will incorrectly solve quadratic equations and will incorrectly graph upside-down parabolas . . . unless you *help* your Apple, by typing parentheses.

To help your Apple, type parentheses whenever a minus sign is followed by an exponentiated number:

```
PRINT -(5^2)
```

Another example:

```
10 X=5
20 PRINT -(X^2)
```

Those parentheses make the Apple print the correct answer, which is 25.

COMMODORE

SCOPE

Commodore makes four computers. The cheapest is the VIC-20; then comes the Commodore 64; then comes the PET; and the most expensive is the CBM. This section explains them all: it explains the VIC, the 64, the PET, and the CBM.

THE KEYBOARD

On the VIC and 64, the zero key is Swedish. On the PET and CBM, the zero key is *not* Swedish.

On some versions of the PET and CBM, the dollar sign is on a separate key from the digit 4, and doesn't require the SHIFT key.

The BACKSPACE key is marked DEL (which stands for DELETE). So to backspace, make sure you hit the DEL key. Do *not* hit the key that shows a left-arrow; the left-arrow key does *not* backspace correctly.

GET STARTED

If you have a VIC or 64, attach it to a TV or monitor. (If you have a PET or CBM, your computer has a monitor built in.)

Flip the computer's power switch to the ON position. (If you have a VIC or 64, the power switch is on the computer's right side. If you have a PET or CBM, the power switch is on the computer's rear.)

If you're using a VIC or 64, turn on the TV or monitor.

If you're using a TV, turn it to channel 3 or 4. To switch from channel 3 to channel 4 (or back to channel 3), flip your computer's channel-selection switch. (The VIC's channel-selection switch is on the unmarked black box that goes between the VIC and the TV. The 64's channel-selection switch is on the back of the 64.) If the writing on the TV looks fuzzy, adjust the TV's fine tuning.

To further improve the appearance of the writing on the screen, adjust the screen's brightness and contrast. (For the PET and CBM, the brightness/contrast knob is on the back of the monitor.)

When the computer is on and functioning correctly, the computer will say READY (instead of OK).

LONG COMMANDS ON SMALL SCREENS

A command can contain up to 80 characters. If you try to type more than 80 characters in a command, the computer will ignore the extra characters.

HOW TO PRINT ON PAPER

Make sure the printer is plugged in, turned on, full of paper, and ready on-line.

To activate the printer, type this:

```
OPEN 4,4: CMD 4
```

That makes the computer use paper instead of your screen. Henceforth, the computer will print all its answers on paper instead of on your screen.

If you ever want the computer to print on your screen again, you must deactivate the printer, by typing:

```
PRINT#4: CLOSE 4
```

For example, to make the computer print just "I LOVE YOU" on paper, you must activate the printer, then tell the computer to print "I LOVE YOU", then deactivate the printer, like this:

```
OPEN 4,4: CMD 4
PRINT "I LOVE YOU"
PRINT#4: CLOSE 4
```

If a program prints on the screen but you want it to print on paper instead, activate the printer before you type RUN. Here's how:

```
OPEN 4,4: CMD 4
RUN
PRINT#4: CLOSE 4
```

Here's a fancy program that prints *two* poems. It prints "LOVE IS LIKE A DOVE" on your screen, and prints "LIFE IS FULL OF STRIFE" on paper:

```
NEW
1 PRINT "LOVE"
2 PRINT "IS LIKE A DOVE"
3 OPEN 4,4: CMD 4
4 PRINT "LIFE"
5 PRINT "IS FULL OF STRIFE"
6 PRINT#4: CLOSE 4
RUN
```

REARRANGING YOUR PROGRAM

To LIST your program on paper instead of on your screen, type this:

```
OPEN 4,4: CMD 4
LIST
PRINT#4: CLOSE 4
```

Unfortunately, your computer does *not* understand the word DELETE. To delete lines 30, 31, 32, 33, and 34, you can*not* type DELETE 30-34; instead, you must type:

```
30
31
32
33
34
```

CLEARING THE SCREEN

The CLEAR key is marked "CLR HOME". To clear the screen, hold down the SHIFT key; and while you keep holding down the SHIFT key, press the "CLR HOME" key.

Your computer does *not* understand CLS. To make line 10 clear the screen, type this instead:

```
10 PRINT "♡";
```

To make that heart, hold down the SHIFT key with the CLR HOME key.

GO TO

The BREAK key is marked "RUN STOP". So to abort your program, press the "RUN STOP" key.

If the computer is printing on the screen too quickly for you to read, you can make the computer print more slowly. To slow the printing, hold down a special key while the computer prints. On the VIC and 64, that special key is marked CTRL; on the PET and CBM, that special key is marked RVS. The computer will print slowly, as long as you keep holding down that special key. When you lift your finger from that special key, the computer will print quickly again.

NUMERIC VARIABLES

A numeric variable can be a letter, or a *pair* of letters, or a letter-followed-by-a-digit. For example, in this program the variable is a pair of letters (MA):

```
10 MA=8
20 PRINT MA+1
```

That program makes the computer print 9.

Any pair of letters is okay, except the following pairs (which have special meanings): GO, TO, IF, ON, FN, TI, ST, OR.

STRING VARIABLES

A string variable looks like a numeric variable, but has a dollar sign after it. For example, you can have a string variable such as MA$.

STRING INPUT

In the INPUT statement, the computer's question is called the *prompt*. The prompt must be short. For the PET and CBM, the prompt must be shorter than 70 characters; for the 64, the prompt must be shorter than 40 characters; for the VIC, the prompt must be shorter than 22 characters.

Look at this example:

```
10 INPUT "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO";C$
```

In that example, the prompt is WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO. That prompt, which contains 45 characters, is too long for the 64 and VIC to handle. If you try to RUN that example, your 64 or VIC will go completely nutty and print lots of error messages, until you cancel the example (by typing NEW).

So if you have a 64 or VIC, you must shorten the prompt. For example, if you have a VIC, you must shorten the prompt to this:

```
10 INPUT "WHAT COLLEGE ADMIT TO";C$
```

Instead of merely shortening the prompt, you can eliminate the prompt altogether and replace it by a PRINT statement, like this:

```
10 PRINT "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO?"
11 INPUT C$
```

That pair of statements works on all the Commodores: even the VIC and 64! It's the only way to make a VIC or 64 print a long question.

Many examples in this book contain prompts that are too long for the VIC and 64 to handle. You must revise the examples. (Or buy a better computer!)

On Commodore computers, the INPUT statement has one other peculiarity: if you try to abort the program while the computer is doing an INPUT statement, the computer will refuse to abort. Instead, you must give a special kind of abortion, called an *input abortion*. Here's how to give an input abortion. On the PET and CBM, just press the RETURN key; on the VIC and 64, hold down the RUN STOP key while also holding down the RESTORE key.

STORIES

In the story examples, some of the prompts are too long for the VIC and 64 to handle, so you must revise them.

BILLS

In the bill example, the prompt in line 10 is too long for the VIC to handle.

NUMERIC INPUT

In the numeric-input examples, some of the prompts are too long for the VIC and 64 to handle.

CONVERSION

In the example about degrees Celcius, the prompt in line 10 is too long for the VIC to handle.

COLONS

Although you can put several statements on the same line, the total number of characters in the line must not be greater than 80.

IF . . . THEN

In line 10 of the example, the INPUT prompt is too long for the VIC to handle.

From now on, I won't continue to remind you to check the length of each prompt. To be safe, get in the habit of rewriting *every* INPUT statement, by making the prompt shorter or by using putting the prompt into a PRINT statement instead.

If you try to invent your own examples using IF . . . THEN, make sure the INPUT prompts are short. Also, make sure each line of your program contains no more than 80 characters. (You'll be tempted to make some of the IF . . . THEN lines be longer than 80 characters, but you must *not* make them longer!)

DISKS

How to turn on the equipment

To be super-safe, remove any disks from the drive. Then turn on the drive and the computer, by flipping their power switches. (If you have a VIC or 64, Commodore recommends that your turn on the drive before the computer. If you have a PET or CBM, Commodore recommends that you turn on the computer before the drive.)

If you have a VIC, type this:

```
OPEN 15,8,15,"UI-"
CLOSE 15
```

How to insert a disk

To put a disk into the drive, first make sure the disk drive's red light is off. Then open the drive's door. (If your drive has *two* doors, use the door on the right.)

Insert the disk into the drive. When you insert it, make sure the disk's label is on *top* of the disk, and make sure the disk's big oval cutout goes into the drive *before* the label does.

Close the drive's door.

How to turn a blank disk into a Commodore disk

If you buy a blank disk and want your computer to use it, you must turn the blank disk into a Commodore disk.

To do that, insert the blank disk into the drive (and close the door). To name the disk ELIZABETH and give it ID number 00, type the following:

PET and CBM

```
HEADER "ELIZABETH",I00
```

VIC and 64

```
OPEN 15,8,15,"N:ELIZABETH,00"
CLOSE 15
```

Each disk you create must have a different ID number. The ID number can be any number from 00 to 99—or it can even be a pair of *letters*.

The drive's red light will glow for about 2 minutes. (The exact time depends on which model drive you have.) When the light stops glowing, your blank disk is finally a Commodore disk.

Commodore disk commands

Commodore's disk commands differ from other computers:

Most computers	PET and CBM	VIC and 64
SAVE "JOE"	DSAVE "JOE"	SAVE "JOE",8
LOAD "JOE"	DLOAD "JOE"	LOAD "JOE",8
RUN "JOE"	DLOAD "JOE" RUN	LOAD "JOE",8 RUN
FILES	CATALOG D0 *this is zero*	LOAD "$",8 LIST
DELETE "JOE"	SCRATCH "JOE"	OPEN 15,8,15,"S:JOE" CLOSE 15

Your program's name can be short (such as JOE) or long: up to 16 characters.

Whenever you give a disk command, the computer sends the command to the disk drive and immediately says READY. The word READY does *not* mean the drive's finished the command yet. The drive doesn't finish the command until the drive's red light stops glowing.

Errors

If you give a wrong disk command, the computer nevertheless sends it to the drive and says READY. But when the drive receives that command, the drive gripes by blinking its red light. The drive's light goes on, then off, then on, then off, then on, etc.

To end that blinking, type this, on most Commodore computers:

```
OPEN 15,8,15,"I"
CLOSE 15
```

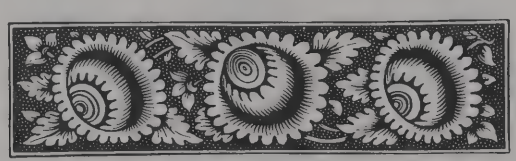
If you have VIC, type this instead:

```
OPEN 15,8,15,"UI-"
PRINT#15, "I"
CLOSE 15
```

If you have a VIC or 64, and try to copy ELIZABETH onto a disk (by saying SAVE "ELIZABETH",8), and the disk *already* contains a program named ELIZABETH, the drive will gripe. To avoid such griping, give this command instead:

```
SAVE "@:ELIZABETH",8
```

In that command, the at-sign followed by the zero tell the drive: save ELIZABETH, *even though the disk already contains an ELIZABETH.*



How to copy a program onto a tape

After positioning the tape and noting the counter's number, do the following . . .

Type the word SAVE; then press the RETURN key. If you have a VIC or 64, your computer will say:

```
PRESS RECORD & PLAY ON TAPE
```

If you have a PET or CBM, your computer will say this instead:

```
PRESS PLAY & RECORD ON TAPE #1
```

Obey the computer, by pressing the recorder's RECORD and PLAY buttons at the same time as each other.

If you have a VIC or 64, your computer will say:

```
OK
SAVING
READY.
```

(Instead of saying SAVING, a PET or CBM says WRITING.)

Then press the recorder's STOP button.

How to read a program from a tape

Put the tape into the recorder. (To do that, you might have to press the recorder's EJECT button.)

By using the REWIND, FAST FORWARD, PLAY, and STOP buttons (and the counter, if you have one), position the tape to the program's beginning (or slightly before).

Type the word LOAD; then press the RETURN key. The computer will say:

```
PRESS PLAY ON TAPE
```

Obey the computer, by pressing the recorder's PLAY button. The computer will say:

```
OK
SEARCHING
FOUND
LOADING
READY.
```

Press the recorder's STOP button. Type the word LIST or RUN; then press the RETURN key.

EDITING

Your screen shows a blinking square, which is called the *cursor*. You can edit your program, by moving the cursor.

Your keyboard has two cursor keys: they say CRSR on them, and also have arrows on them pointing up, down, left, and right. By pressing those keys, with or without the SHIFT key, you can move the cursor in all four directions. Try it!

To move the cursor far in a direction, press a cursor key repeatedly. Or simply hold the cursor key down for a long time. The longer you hold the cursor key down, the farther the cursor will move.

To edit line 20, do the following. . . .

First, make sure line 20 is on your screen. (If it's not on your screen yet, type LIST 20, which will make it appear.)

Next, move the cursor to where the error is (by using the cursor keys).

Fix the error, by typing over it. If your correction is shorter than the original version, hold down the DEL key, which will delete the unwanted extra letters. If your correction is *longer* than the original version, hold down the SHIFT and INST keys simultaneously, which will insert extra spaces to hold your new, long version; but be careful not to insert more spaces than you really need, since eliminating excess spaces is difficult.

After fixing the error, so that the line on the screen looks correct, press the RETURN key. That registers the new version. (If you forget to press the RETURN key, the computer will ignore your changes, even though they remain on the screen.) Press the RETURN key just once.

After pressing the RETURN key once, use the CRSR down-arrow key to move down to the bottom of the screen, so that you can work on your next problem. Congratulations: you've finished the edit!

That editing procedure is easy, except for one hassle: under certain unusual conditions, the cursor keys and INST/DEL key don't work! Under those conditions, the cursor keys and INST/DEL keys print strange symbols on your screen instead of moving the cursor. If that happens to you, press the RETURN key and then try again to edit the line. If it happens again, press the RETURN key, then move the cursor to the bottom of the screen, and type the whole line over again.

I said that the hassle occurs "under certain unusual conditions". Specifically, it occurs only if you begin an editing sequence (by pressing the cursor keys and INST/DEL key) when you're at the right-hand part of a line: the part that's to the right of an unmatched quotation mark or to the right of an inserted space (inserted by the INST key).

RENUMBERING

Commodore computers don't understand the word RENUM.

PAUSING

Commodore's keyboard lacks a PAUSE key. Instead of pausing, you must either slow the computer down (by holding down the CTRL or RVS key) or abort (by pressing the STOP key).

After aborting a RUN, you can make the computer continue where it left off, by typing:

```
CONT
```

The word CONT will make the computer continue an aborted RUN, but will *not* make it continue an aborted LIST.

APOSTROPHE

Instead of an apostrophe, type a colon followed by the word REM (which stands for "remark"). For example:

```
40 C=30: REM BECAUSE RUSS HAS 30 COMPUTERS
```

LISTS OF DATA

If you invent your own list of data, remember that each line of your program must be short: no more than 80 characters!

COUNTING TO 100

If you indent the lines between FOR and NEXT, your computer ignores the indentation. When you LIST your program, you won't see any indentation. So don't bother to indent.

ZONES

The VIC has just 2 zones; each zone is 11 characters wide (so the entire screen is 22 characters wide). The 64 and PET and old CBM have 4 zones; each zone is 10 characters wide (so the entire screen is 40 characters wide). The new CBM has 8 zones; each zone is 10 characters wide (so the entire screen is 80 characters wide).

TAB

On other computers, the leftmost character in the line is said to be at position 1. But on Commodore computers, the leftmost character in the line is said to be at *position 0* instead; after position 0 come positions 1, 2, 3, etc.

So on Commodore computers, if you say TAB(0), you'll be going to the leftmost position; if you say TAB(1), you will *not* be going to the leftmost position.

CALENDAR

The VIC's screen is too narrow to handle the calendar.

LOCATE

Commodore computers don't understand the word LOCATE. To make Commodore computers move the cursor, use the following trick instead. . . .

In the middle of your program, write a PRINT statement that prints a string. When you're typing that PRINT statement, remember to put the string in quotation marks: so type the word PRINT, followed by a quotation mark, and then start typing the string.

When you're typing the string, press a cursor key several times in the middle of the string. When you do that, you'll see funny symbols in the middle of your PRINT statement.

When you RUN the program, and the computer comes to the PRINT statement, and comes to those funny symbols, the computer will automatically move the cursor.

RANDOM NUMBERS

For random numbers, Commodore's notation differs from other computers.

For a random integer from 1 to 5, other computers use RND(5); Commodore uses 1+INT(RND(1)*5) instead.

For a random decimal between 0 and 1, other computers use RND; Commodore uses RND(1) instead.

To begin using random numbers, other computers use RANDOMIZE. If you have a VIC or 64 or CBM, use R=RND(0) instead. If you have a PET, use R=RND(-TI) instead.

PIXELS

Commodore computers can't easily handle pixels.

MUSIC

The PET and CBM can't play music. The 64 can play music, but the method is too complicated to be practical.

The VIC, on the other hand, has a pleasantly simple way to play music. Here's how to play music on the VIC. . . .

Four electronic people, who sing, are hiding inside your VIC. They're called the four *voices*.

Voice 1 is the bass; voice 2 is the tenor; voice 3 is the soprano; and voice 4 is the comedian who sounds like a thunderstorm.

Voice 1 is called V+1. Voice 2 is called V+2. Voice 3 is called V+3. Voice 4 is called V+4. The volume they all sing at is called V+5.

To create music, begin by typing this:

```
V=36873
```

Then set the volume (V+5) to any number from 0 to 15. (0 is silence; 15 is the loudest and also the most distorted; most people prefer 4.) For example, to set the volume to 4, type this:

```
POKE V+5,4
```

(If you'd rather set the volume to 15, type POKE V+5,15 instead.)

You can set each voice to a pitch from 128 to 254. (128 is a low note; 254 is a high note.) For example, to set voice 2 to a pitch of 150, type this:

```
POKE V+2,150
```

That makes voice 2 "sing" pitch 150. It sings through your TV. Adjust your TV's volume knob.

By giving several POKE statements, you can make all the voices play simultaneously. (But you'll probably want to avoid voice 4, which sounds like a thunderstorm.) If you tell a voice to sing a pitch, it will continue to sing that pitch until you give it another pitch to sing.

If you want a voice to sing *nothing* (and shut up!), tell it to sing pitch 0. For example, to make voice 2 shut up, type this:

```
POKE V+2,0
```

Here's an easier way to make all the voices shut up: while you hold down the STOP key, tap the RESTORE key.

You can put those commands into a program, like this:

```
Get ready for music.                    10 V=36873
Set the volume (V+5) to medium-low (4). 20 POKE V+5,4
Set voice 1 to a pitch of 128.          30 POKE V+1,150
Set voice 2 to a pitch of 160.          40 POKE V+2,200
Set voice 3 to a pitch of 192.          50 POKE V+3,250
Set voice 4 (thunderstorm) to 224.      60 POKE V+4, 254
Wait, while the voices keep playing.    70 FOR I = 1 TO 1000: NEXT
Make all four voices shut up.           80 FOR I = 1 TO 4: POKE V+I,0: NEXT
```

To make that example sound better, omit the thunderstorm (line 60).

I said that the pitch can be any number from 128 to 254. Here are their musical equivalents: 128=B, 135=C, 142=C#, 148=D, 154=D#, 160=E, 165=F, 170=F#, 175=G, 179=G#, 184=A, 188=A#, 191=B, 195=C, 198=C#, 202=D, 205=D#, 207=E, 210=F, 213=F#, 215=G, 217=G#, 219=A, 221=A#, 223=B, 225=C, 227=C#, 228=D, 230=D#, 231=E, 233=F, 234=F#, 235=G, 236=G#, 237=A, 238=A#, 239=B, 240=C, 241=C#, 242=D, 243=E, 244=F, 245=G, 246=A, 247=B, 248=C#, 249=E, 250=G, 251=B, 252=E, 253=B, 254=B.

That list is accurate. When you bought your VIC, Commodore gave you a manual; but the list in Commodore's manual is full of errors! Use my list, not Commodore's. My list is mathematically accurate (it came from a physics formula); the list in Commodore's manual was developed by a non-physicist who was also half-deaf.

QUIZ

If you have a VIC (which can't handle long input prompts), omit the semicolon at the end of line 110.

SCOPE

This section explains the IBM Personal Computer (which is called the IBM PC). The information in the section applies to the original version of the IBM PC and also to the fancier version (which is called the IBM PC XT).

THE KEYBOARD

The zero key isn't Swedish.

Your keyboard has two SHIFT keys. The left SHIFT key is above the ALT key. The right SHIFT key is above the CAPS LOCK key. Each SHIFT key has an up-arrow on it (instead of the word SHIFT).

The letters are *not* automatically capitalized. If you press the R key, you'll get a small r. To get a capital R, you must hold down the SHIFT key.

The BACKSPACE key is *to the left of the NUM LOCK key*. The BACKSPACE key has a left-arrow on it.

The RETURN key's above the PRTSC key. The RETURN key has a bent arrow on it.

GET STARTED

To start, you can choose either the "easy" way or the "fancy" way. The "easy" way is easier to understand; the "fancy" way lets your IBM do extra tricks.

If you choose the "easy" way, do *not* use a disk; remove any disks from the drives.

If you choose the "fancy" way, you need a disk drive, and you must make sure it contains a special kind of disk. The disk should be a "DOS system master" (which contains BASIC) or a copy of that disk. Put the disk into the *left* disk drive. When you put the disk into the drive, make sure the disk's label is on *top* of the disk, and make sure the disk's big oval cutout goes into the drive *before* the label does. Close the drive's door.

Attach the computer to a TV or monitor. Turn on the TV or monitor. (If you're using a TV, turn it to channel 33.) Then flip the computer's power switch to the ON position. (The power switch is red; it's on the right side of the computer.)

If you put your ear next to the computer, you'll hear its fan hum softly.

At first, the screen will display strange blips, as if it were broken. But after 8 seconds, the screen will become black, except for one blinking horizontal line in the upper left corner.

After about 30 seconds (the exact time depends on how much RAM your computer has), the computer will beep. Then the computer will print a message on the screen.

If you're using a TV (instead of a monitor), the message will be hard to read. Try channels 32, 33, 34, and 10; see which channel works best on your TV. Adjust your TV's fine-tuning knob. Also, adjust the contrast and brightness. Even after making all those adjustments, the writing will still look somewhat fuzzy; the computer will improve the writing if you give it a special command, which I'll explain later.

If you used the "easy" way (without disks), skip to the next paragraph. If you used the "fancy" way, the computer will say "Enter new date". (If your "DOS system master" disk is old, the wording might be slightly different.) Type the date. For example, if today is January 27, 1984, type 01-27-84 and then press the RETURN key. The computer will say "Enter new time". Type the time, using a 24-hour clock. For example, if the time is 1:45 AM, type 1:45 and then press the RETURN key; if the time is 1:45 PM, type 13:45 and then press the RETURN key; if the time is 7 seconds after 1:45 PM, type 13:45:07 and then press the RETURN key. The computer will say:

```
A>
```

Type the word "basica", like this:

```
A>basica
```

Then press the RETURN key.

Regardless of whether you used the "easy" method or the "fancy" method, the computer will finally say "Ok".

If you're using a TV instead of a monitor, type:

```
width 40
```

That tells the computer to make the writing twice as wide, so that you can read it more easily.

ARITHMETIC

The computer doesn't care about capitalization. To type the word PRINT, you can use either capital letters or small letters; the computer doesn't care. You'll probably prefer small letters, because they're easier to type, since they don't need the SHIFT key.

QUESTION MARK

Instead of typing the word PRINT, you can type a question mark.

For even more fun, hold down the ALT key; and while you keep holding down the ALT key, tap the P key. The screen will automatically show the word PRINT.

So the funniest way to say PRINT is to type an ALTered P.

PROGRAMMING

If you tap the F2 key, the screen will automatically show the word RUN and will automatically press the RETURN key—so that the computer will automatically run your program.

GRIPES

If the computer gripes at you, press the RETURN key immediately.

After pressing the RETURN key, correct the error, by retyping the line that the computer griped about.

HOW TO PRINT ON PAPER

To print on paper, you can use three methods.

1. The simplest method is to say LPRINT instead of PRINT. For example, to make the computer say "I love you" on paper, type this:

```
lprint "I love you"
```

2. Another simple method is to hold down the SHIFT key; and while you keep holding down the SHIFT key, tap the PRTSC key. That makes the computer print, on paper, a copy of the entire screen.

3. The third method is slightly more advanced; it works only if you're using a disk called "DOS system master version 2.00". Hold down the CTRL key; and while you keep holding down the CTRL key, tap the PRTSC key. That command tells the computer that henceforth, anything that appears on the screen should be copied onto paper. The command remains in effect until you press the CTRL and PRTSC keys simultaneously again.

REARRANGING YOUR PROGRAM

To LIST your program on paper instead of on your screen, type LLIST instead of LIST.

If you tap the F1 key, the screen will automatically say LIST. If you type an ALTered D (by tapping the D key while holding down the ALT key), the screen will automatically say DELETE.

CLEARING THE SCREEN

Your computer lacks a CLEAR key. To clear the screen, hold down the CTRL key; and while you keep holding down the CTRL key, tap the HOME key.

Another way to clear the screen is to give the CLS command.

GO TO

If you type an ALTered G (by tapping the G key while holding down the ALT key), the screen will automatically say GO TO (or GOTO).

The BREAK key is at the upper right-hand corner of the keyboard. The BREAK key says BREAK on it; it also says SCROLL LOCK on it.

The BREAK key works only if you simultaneously hold down the CTRL key. So to abort a program, do this: hold down the CTRL key; and while you keep holding down the CTRL key, tap the BREAK key.

NUMERIC VARIABLES

A numeric variable's name can be a letter (such as X) or a longer combination of characters (such as profit.in.1984.before.November.promotion). For example, you can type:

```
10 profit.in.1984.before.November.promotion = 3497.18
20 profit.in.1984 = profit.in.1984.before.November.promotion + 6214.27
30 print profit.in.1984
```

The computer will print:

```
 9711.45
```

The variable's name can be quite long: up to 40 characters! The first character in the name must be a letter; the remaining characters can be letters, digits, or periods. The name must not be a word that has a special meaning to the computer; for example, the name cannot be PRINT or GO or TO.

STRING VARIABLES

A string variable looks like a numeric variable, but has a dollar sign after it. For example, you can have a string variable such as:
my.job.in.1984.before.November.promotion$

STRING INPUT

If you type an ALTered I, the screen will automatically say INPUT.

IF . . . THEN

If you type an ALTered T, the screen will automatically say THEN.

DISKS

How to prepare the computer for handling disks

At the beginning of this IBM appendix, you'll find a section entitled "GET STARTED". It explains how to get started using the computer. It says there are two ways: an easy way, and a fancy way.

To use disks, you must start the computer by using the fancy way. For details, reread the section entitled "GET STARTED".

If you started the computer by using the easy way instead, turn the computer off and then restart the computer by using the fancy way.

How to turn a blank disk into an IBM disk

If you buy a blank disk and want the IBM computer to use it, you must turn the blank disk into an IBM disk.

To do that, put the blank disk into the disk drive that's on the right. Then type the following:

```
system
diskcopy a: b:
```

The computer will say:

```
Insert source diskette into drive A:
Insert target diskette into drive B:
Strike any key when ready
```

Press any key on the keyboard. The computer will copy all information from the left drive's disk to the blank disk. The blank disk will become an IBM disk that's an exact copy of the other disk.

Then the computer will ask:

```
Copy another (Y/N)?
```

If you don't want to make another copy, type the letter n.

Finally, return to BASIC, by typing:

```
basica
```

IBM disk commands

The following commands all work, without any hassles: SAVE, FILES, LOAD, and RUN.

The only command that's peculiar is KILL. To erase JOE from the disk, you can*not* say KILL "JOE". Instead, you must say:

```
kill "joe.bas"
```

The "bas" emphasizes that JOE was written in BASIC.

Your program's name can be short (such as JOE) or long: up to 8 characters. The name's first character must be a letter; the remaining characters should be letters or digits.

The left drive is called *drive A*; the other drive is called *drive B*. If you want JOE to be in drive B instead of in drive A, put a B and a colon before JOE, like this:

```
save "b:joe"
```

To make the computer print the names of all the programs in drive B, type this:

```
files "b:*.*"
```

If your computer's new (and uses DOS version 2 instead of DOS version 1), you can omit the asterisks and the period; just type:

```
files "b:"
```

TAPES

Practically everybody who buys an IBM Personal Computer uses disks instead of tapes. Therefore, I won't bother explaining IBM's tape system, which practically nobody uses.

EDITING

To edit line 20, type:

```
edit 20
```

The computer will reprint line 20 and put you at the beginning of that line.

Look at line 20, and find the error you want to correct. Move to where the error begins, by pressing the right-arrow key several times. (The right-arrow key is at the far right side of the keyboard, and also has a 6 on it.)

Delete the error, by pressing the DEL key several times. (The DEL key is at the lower right corner of the keyboard.)

Press the INS key once: it tells the computer that you want to insert new characters. (The INS key is next to the DEL key.)

Type the new characters you want to insert. As you type them, the other characters on the line automatically move out of the way, to make room for the new characters you're typing.

When the line looks perfect, press the RETURN key.

PAUSING

To make the computer pause, hold down the CTRL key; and while you keep holding down the CTRL key, tap the NUM LOCK key.

To stop pausing, tap any other key.

COUNTING TO 100

To indent, you can hit the space bar repeatedly. But to indent more easily, hit the TAB key (which is above the CTRL key and has a right-arrow on it).

ZONES

If you're using a monitor that prints 80 characters per line, your screen has five zones: zones 1-4 are each 14 characters wide; zone 5 is 24 characters wide.

If you said WIDTH 40 (because you're using a TV instead of a monitor), your screen has just two zones: zone 1 is 14 characters wide; zone 2 is 26 characters wide.

RANDOM NUMBERS

Instead of saying just RANDOMIZE, you must say something longer. If your computer's new (and uses DOS version 2 instead of DOS version 1), say RANDOMIZE TIMER. If your computer's old (and uses DOS version 1 instead of DOS version 2), say RANDOMIZE VAL(RIGHT$(TIME$,2)).

For a random integer between 1 and 5, say 1+INT(RND*5) instead of RND(5).

PIXELS

Screens

Your computer can handle three kinds of screens. If you say—

```
screen 1
```

you'll be using medium-resolution graphics, for which the X coordinate goes from 0 to 159, and for which the Y coordinate goes from 0 to 199. If you say SCREEN 2 instead of SCREEN 1, you'll be using high-resolution graphics, for which the X coordinate goes from 0 to 639, and for which the Y coordinate still goes from 0 to 199. If you say SCREEN 0, you'll return to non-graphics.

If you're a beginner, say SCREEN 1 instead of SCREEN 2, because SCREEN 1 creates fatter dots (so you can see them better), and because SCREEN 1 is more fun (it lets you play with colors). So if you're a beginner, begin your graphics experience by saying SCREEN 1.

Fundamental shapes

Your computer can draw three fundamental shapes: dots, lines, and circles. To make the computer draw a dot at pixel (100,100), say:

```
pset (100,100)
```

To make the computer draw a line from pixel (0,0) to pixel (100,100), say:

```
line (0,0)-(100,100)
```

To make the computer draw a circle whose center is pixel (100,100) and whose radius is 50, say:

```
circle (100,100),50
```

When you draw those shapes, the computer draws them in white, on a black background.

If you wish, you can put those commands into a program. Your program's top line should say SCREEN 1 (or SCREEN 2), which erases all previous shapes from the screen.

Painting

After you've drawn an outline of a shape (by using dots, lines, and circles), you can fill in the middle of the shape, by telling the computer to PAINT the shape.

Here's how to PAINT a shape that you've drawn (such as a circle or a house). Find a pixel that's in the middle of the shape and that's still black; then tell the computer to PAINT, starting at that pixel. For example, if pixel (100,101) is inside the shape and is still black, say:

```
paint (100,101)
```

Colors

In medium resolution (SCREEN 1), you can use four colors: color 0 is black; color 1 is cyan (greenish blue); color 2 is magenta (purplish red); and color 3 is white. In high resolution (SCREEN 2), you have only two colors: color 0 is black, color 1 is white.

Normally, the PSET, LINE, CIRCLE, and PAINT commands draw in white. If you prefer a different color, put a comma and the color's number at the end of the command. For example, if you're using medium resolution and want to draw a line from (0,0) to (0,100) in magenta (whose color number is 2), type this:

```
line (0,0)-(0,100),2
```

When you give a PAINT command, you must make its color the same as the color of the outline you're filling in.

Because American TV's are crummy, you'll get the correct colors only if you draw horizontal lines. If you draw vertical lines or diagonals or circles or dots, the colors will be slightly off. (That's because, on your TV, a pixel whose X coordinate is odd can't be cyan, and therefore appears reddish; conversely, a pixel whose X coordinate is even can't be red, and therefore appears cyanish.)

The horizontal lines will be perfect colors. If they don't look perfect to you, adjust your TV's COLOR and TINT dials.

Changing the background

Normally, your drawing has a black background. But if you say—

```
color 1
```

everything on the screen that was black suddenly becomes blue. Moreover, any future drawing you try to do in black will be done in blue instead. That's because, in the COLOR command, 1 is the code number for blue. To choose your background, use these numbers: 0=black, 1=blue, 2=green, 3=cyan (greenish blue), 4=red, 5=magenta (purplish red), 6=brown, 7=white.

If you put an "8+" in front of the background's number, you'll get a lighter background. For example, if you say—

```
color 8+1
```

the background becomes *light* blue.

If you ask for "light black", you'll get gray. If you ask for "light cyan", you'll get aqua. If you ask for "light red", you'll get pink. If you ask for "light brown", you'll get yellow. If you ask for "light white", you'll get super-bright high-intensity white.

Changing the palette

Normally, your drawing has a black background, and uses a *palette* consisting of three colors: color 1 is cyan, color 2 is magenta, and color 3 is white. That's called palette #1. If you wish, you can switch to palette #0, which consists of a different three colors: color 1 is green, color 2 is red, and color 3 is brown. To switch to palette #0, say:

```
color ,0
```

Remember to put the comma before the zero.

When you type that command, everything that was colored using palette #1 (cyan, magenta, white) suddenly changes to palette #0 (green, red, brown). In other words, everything that was cyan suddenly becomes green; everything that was magenta suddenly becomes red; and everything that was white suddenly becomes brown. If you do any more drawing afterwards, you'll be using palette #0: color 1 is now green, color 2 is now red, and color 3 is now brown.

To return to palette #1, say:

```
color ,1
```

You can define the background color and the palette at the same time, like this:

```
color 8+1,0
```

That makes the background color be 8+1 (light blue) and makes the palette be #0 (green, red, brown).

Boxes

If you type—

```
line (0,0)-(100,100),2
```

the computer draws a line from pixel (0,0) to (100,100) using color 2.

If you put the letter B at the end of the line command, like this—

```
line (0,0)-(100,100),2,b
```

the computer will draw a box instead of a line. One corner of the box will be at pixel (0,0); the opposite corner will be at (100,100); and the box will be drawn using color 2.

If you put BF at the end of the line command, like this—

```
line (0,0)-(100,100),2,bf
```

the computer will draw a box and also fill it in. In other words, the computer will draw the box and then automatically PAINT its interior.

MUSIC

To produce sounds, IBM provides 3 commands: BEEP, SOUND, and PLAY. BEEP appeals to business executives; SOUND appeals to doctors and engineers; and PLAY appeals to musicians. Let's look at each of those words.

How to BEEP

If you type—

```
beep
```

the computer will beep. Specifically, it will play a note whose frequency ("pitch") is 800 hertz, and it will play the note for a quarter of a second.

You can say BEEP in the middle of your program. For example, you can tell the computer to BEEP if a person enters wrong data. In the following example, the computer acts as a priest; it performs a marriage ceremony:

```
10 input "Do you take this woman to be your lawful wedded wife";a$
20 if a$<>"I do" then beep: print "Try again!": go to 10
30 input "Do you take this man to be your lawful wedded husband";a$
40 if a$<>"I do" then beep: print "Try again!": go to 30
50 print "I now pronounce you man and wife."
```

Line 10 makes the computer ask the groom, "Do you take this woman to be your lawful wedded wife?" If the groom doesn't say "I do", line 20 makes the computer beep and say "Try again!" and repeat the question. Lines 30 and 40 do the same thing to the bride. Line 50 congratulates the couple for having answered correctly.

How to SOUND

If you type—

```
sound 440,18.2
```

the computer will produce a sound. In that command, the 440 is the frequency ("pitch"), measured in hertz (cycles per second); so the sound will be a musical note whose pitch is 440 hertz. (That note happens to be "the A below middle C".)

If you replace the 440 by a higher number, the sound will have a higher pitch. The pitch can be any number from 37 to 32767; but 32767 is higher than you can hear.

When you were a baby, you could probably hear up to 20000; as you get older, your hearing gets worse, and you can't hear such high notes. Today, the highest sound you can hear is probably somewhere around 15000.

To find out, give yourself a hearing test, by running this program:

```
10 print "What pitch would you like me to play";p
20 sound p,18.2
30 go to 10
```

When you run that program, begin by inputting a low pitch (such as 37). Then input a higher number, then an even higher number, until you finally pick a number so high you can't hear it. (When trying that test, put your ear close to the computer's speaker, which is in the computer's front left corner.) When you've picked a number too high for you to hear, try a slightly lower number. Keep trying different numbers, until you find the highest number you can hear.

I'm 36 years old, and the highest pitch *I* can hear is about 14500. How about *you*? Is your hearing better than mine?

Have a contest with your friends: find out which of your friends can hear best.

If you run that program every year, you'll see that your hearing gets gradually worse.

In those examples, the 18.2 makes the computer produce the sound for 1 second. If you want the sound to last longer—so that it lasts 2 seconds—replace the 18.2 by 18.2*2. For 10 seconds, say 18.2*10. (That's because the computer's metronome beats 18.2 times per second.)

How to PLAY

At the beginning of the IBM appendix is a section entitled "GET STARTED". It says you can start using the IBM computer in two ways: the fancy way (which requires disks) or the easy way. To use the PLAY command, you must choose the fancy way.

If you choose the fancy way, you can use the PLAY command and can type this:

```
play "cdg#b-a"
```

The computer will play the note C, then D, then G sharp, then B flat, then A.

The computer can play in seven octaves, numbered from 0 to 6. Octave 0 consists of very bass notes; octave 6 consists of very high-pitched notes. In each octave, the lowest note is a C: the notes in an octave are C, C#, D, D#, E, F, F#, G, G#, A, A#, and B.

"Middle C" is at the beginning of octave 3. Normally, the computer plays in octave 4. To make the computer switch to octave 3, type the letter "o" followed by a 3, like this:

```
play "o3"
```

After giving that command, anything else you PLAY will be in octave 3, until you change octaves again.

If your IBM is new (and uses DOS version 2 instead of DOS version 1), you can use the symbol ">" to mean "go up an octave", and you can use the symbol "<" to mean "go down an octave". For example, if you say—

```
play "g>cd<g"
```

the computer will play the note G, then go up an octave to play C and D in that higher octave, then go down to the original octave to play G again.

The lowest note the computer can play (which is the C in octave 0) is called *note 1*. The highest note the computer can play (which is the B in octave 6) is called *note 84*. To make the computer play note 84, type this:

```
play "n84"
```

To make the computer play its lowest note (1), then its middle note (42), then its highest note (84), type this:

```
play "n1n42n84"
```

Besides playing with pitches, you can also play with rhythms. Normally each note is a "quarter note"; if you want an eighth note instead (which is faster), put an 8 after the note's name, like this:

```
play "ce8f#8gc"
```

The computer will play C (as a normal quarter note), then E faster (as an 8th note), then F# fast also (as an 8th note), then G and C normally (as quarter notes).

If you say L8, the following notes will all be 8th notes—until you give another L command. This plays the same music as the previous example:

```
play "cl8ef#l4gc"
```

It plays a normal C, then some 8th notes (E and F#), then some quarter notes (G and C).

Besides using 8 for eighth notes, you can use 16 for sixteenth notes (which are even faster), 32 for thirty-second notes (which are super-fast), and 64 for sixty-fourth notes (which are super-super-fast). For long notes, you can use 2 (which gives a half note) or 1 (which gives a whole note). You can use any number from 1 to 64. You can even use in-between numbers, such as 7 or 23 (though such rhythms are hard to stamp your foot to).

If you put a period after the rhythm number, the computer will multiply the duration by 1½. For example, if you say "8." instead of just "8", the computer will play the note 1½ times as long as an 8th note. Musicians call that "a dotted eighth note."

To make the computer pause ("rest") for an eighth note, put a P8 into the string.

Normally, the computer plays 120 quarter notes per minute; but you can change that tempo. To switch to 150 quarter notes per minute, say:

```
play "t150"
```

You can switch to any tempo from 32 to 255. The 32 is very slow; 255 is very fast. In musical terms, 40=larghissimo, 50=largo, 63=larghetto, 65=grave, 68=lento, 71=adagio, 76=andantino, 92=andante, 114=moderato, 120=allegretto, 144=allegro, 168=vivace, 188=presto, and 208=prestissimo.

EXPONENTS

For exponents, IBM uses the symbol "^" instead of "↑". To type the symbol "^", hold down the SHIFT key; and while you keep holding down the SHIFT key, tap the 6 key that's in the top row.

RADIO SHACK

SCOPE

Radio Shack makes many computers, which use many versions of BASIC. Radio Shack's *best* version of BASIC is called *model 4 BASIC*. Two other popular versions (which are *almost* as good) are *model 3 BASIC* and *extended color BASIC*. This section explains those versions of BASIC.

To get model 4 BASIC (which is the best), you must buy a TRS-80 model 4 computer that has disk drives. To get model 3 BASIC, you must buy a TRS-80 model 4 or model 3 computer; you don't need disk drives. To get extended color BASIC, you must buy a TRS-80 Color Computer 1 or a TRS-80 Color Computer 2.

When you buy a computer from Radio Shack, make sure you get one of those three versions of BASIC! For example, if you buy a TRS-80 model 3 computer, make sure you get model 3 BASIC, *not* level 1 BASIC (which costs less but is crummy). If you buy a TRS-80 Color Computer 1 or 2, make sure you get *extended* color BASIC, not *standard* color BASIC (which costs less but is crummy).

THE KEYBOARD

The zero key is Swedish.

In model 3 BASIC and extended color BASIC, the letters are automatically capitalized. In model 4 BASIC, the letters are *not* automatically capitalized: to capitalize, you must hold down the SHIFT key.

The BACKSPACE key has a left-arrow on it. So if you press the left-arrow key, the computer will erase the last character you typed.

If you press the left-arrow key and the SHIFT key simultaneously, the computer will erase the entire *line* you've been typing.

The RETURN key says ENTER on it.

GET STARTED

The way to get started depends on which version of BASIC you're using.

How to start extended color BASIC. Attach the computer to a TV. Turn on the computer's power, by pressing the POWER button, which is at the computer's rear. Turn on the TV, and turn it to channel 3 or 4. To switch from channel 3 to channel 4 (or back to channel 3), flip your computer's channel-selection switch, which is at the computer's rear. When the computer is on and functioning correctly, the computer will say OK.

How to start model 4 BASIC. Remove any disks from the disk drives. Flip on the computer's power switch, which is underneath the computer's right side. The red light on the bottom disk drive will turn on, then off. Insert a disk that contains "Model 4 TRSDOS and BASIC Interpreter" into the bottom drive; when you insert it, make sure the disk's label is on *top* of the disk, and make sure the disk's big oval cutout goes into the drive *before* the label does. Close the drive's door. On the far right side of the keyboard, you'll see an orange button; press it. The computer will say "Date"; if the writing looks fuzzy, turn the brightness and contrast dials, which are underneath the computer's left side. Type the date; for example, if today is January 27, 1984, type 01/27/84 and then press the ENTER key. The computer will say "TRSDOS Ready". Type this word:

```
basic
```

Then press the ENTER key. After a brief pause, the computer will say "Ready". Notice that the computer says "Ready" instead of "OK".

How to start model 3 BASIC, if you're not planning to use any disks. Hold down the BREAK key; and while you keep holding down the BREAK key, flip on the computer's power switch (which is underneath the computer's right side). The computer will say "Cass"; if the writing looks fuzzy, turn the brightness and contrast dials, which are underneath the computer's left side. Press the ENTER key. The computer will say "Memory Size". Press the ENTER key. The computer will say READY (instead of OK).

How to start model 3 BASIC, if you're planning to use disks. The procedure resembles "how to start model 4 BASIC", but has two differences:

The disk you insert contains "Model 3 TRSDOS" instead of "Model 4 TRSDOS".

After you enter the date (and press the ENTER key), the conversation proceeds as follows. The computer will say "Enter Time". Type the time, by using a 24-hour clock; for example, if the time is 1:45 PM, type 13:45:00 and then press the ENTER key. The computer will say "TRSDOS Ready". Type the word BASIC and then press the ENTER key. The computer will say "How Many Files". Press the ENTER key. The computer will say "Memory Size". Press the ENTER key. The computer will say READY.

ARITHMETIC

In model 4 BASIC, the computer doesn't care about capitalization: so to type the word PRINT, you can use either capital letters or small letters; you'll probably prefer small letters, because they're easier to type.

GRIPES

In extended color BASIC, if the computer gripes at you, just retype the line that the computer griped about.

In model 3 and model 4 BASIC, if the computer gripes at you, *press the ENTER key immediately*; after pressing the ENTER key, retype the line that the computer griped about.

HOW TO PRINT ON PAPER

To make extended color BASIC print on paper, say "PRINT#-2," instead of "PRINT". For example, to make extended color BASIC say "I LOVE YOU" on paper, type this:

```
PRINT#-2, "I LOVE YOU"
```

To make model 3 or model 4 BASIC print on paper, you can use three methods:

1. The simplest method is to say LPRINT instead of PRINT. For example, to make the computer say "I LOVE YOU" on paper, type this:

```
LPRINT "I LOVE YOU"
```

2. Another simple method is to hold down the CTRL key; and while you keep holding down the CTRL key, tap the colon key (which is next to the zero key). That makes the computer print, on paper, a copy of the entire screen. (On the model 3 computer, which lacks a CTRL key, do the following instead: while depressing the down-arrow key and left SHIFT key simultaneously, tap the colon key.)

3. The third method is slightly more advanced. It consists of typing a special command that activates the printer. That command tells the computer that henceforth, anything that appears on the screen should be copied onto paper—until you give another special command that *de*activates the printer. In model 4 BASIC, you activate the printer by saying—

```
system "link *do *pr"
```

and deactivate the printer by saying:

```
system "reset *do"
```

In model 3 BASIC, *you need disk drives*, and you activate the printer by saying—

```
CMD "Z","ON"
```

and deactivate the printer by saying:

```
CMD "Z","OFF"
```

REARRANGING YOUR PROGRAM

To LIST your program on paper instead of on your screen, type LLIST instead of LIST.

Model 3 and model 4 BASIC understand the word DELETE. For extended color BASIC, say DEL instead of DELETE.

CLEARING THE SCREEN

To clear the screen in extended color BASIC and model 3 BASIC, press the CLEAR key. To clear the screen in model 4 BASIC, press the CLEAR key *while holding down the SHIFT key*.

Another way to clear the screen is to type CLS.

GO TO

To abort a program, press the BREAK key.

NUMERIC VARIABLES

The rules about variables depend on which version of BASIC you're using.

Variables in extended color BASIC and model 3 BASIC. A numeric variable's name can be a letter (such as X), or a letter-followed-by-a-digit (such as X2), or a *pair* of letters (such as MA). For example, in this program the variable is a pair of letters (MA):

```
10 MA=8
20 PRINT MA+1
```

That program makes the computer print 9. Any pair of letters (such as MA) is okay, except the following pairs (which have special meanings): GO, TO, IF, ON, FN, OR.

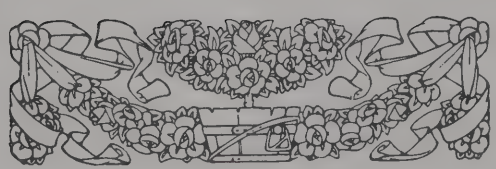
Variables in model 4 BASIC. A numeric variable's name can be a letter (such as X) or a longer combination of characters (such as profit.in.1984.before.November.promotion). For example, you can type:

```
10 profit.in.1984.before.November.promotion = 3497.18
20 profit.in.1984 = profit.in.1984.before.November.promotion + 6214.27
30 print profit.in.1984
```

That program makes the computer print 9711.45. The variable's name can be quite long: up to 40 characters! The first character in the name must be a letter; the remaining characters can be letters, digits, or periods. The name must not be a word that has a special meaning to the computer; for example, the name cannot be PRINT or GO or TO.

STRING VARIABLES

A string variable looks like a numeric variable, but has a dollar sign after it.



STRING INPUT

If you're using model 4 BASIC, string input is easy, so you can skip to the next topic (which is "COUNTING TO 100").

If you're using model 3 BASIC or extended color BASIC, string input is more complicated, because the computer might gripe about your string input. If the computer decides to gripe, it will say—

```
Out of string space
```

or, more briefly:

```
?OS ERROR
```

To stop the computer from griping, insert this statement:

```
1 CLEAR 1000
```

If the computer *still* gives that kind of gripe, raise 1000 to a higher number, like this:

```
1 CLEAR 2000
```

Keep raising, until the computer stops that gripe. On the other hand, if you raise *too* high, the computer will say—

```
Out of memory
```

or, more briefly:

```
?OM ERROR
```

That means you must go lower.

Here's why. . . .

Your program contains two kinds of strings: the strings that are known, and the strings that are unknown. For example, if you say—

```
10 G$="DOWN"
```

the G$ is known to be DOWN. If you say—

```
10 INPUT "WHAT IS YOUR NAME";N$
```

the N$ is unknown, because its value depends on what the human inputs.

Normally, the computer reserves enough memory space to hold just a few characters. Model 3 BASIC makes the computer reserve enough memory space to hold just 50 characters; extended color BASIC makes the computer reserve enough memory space to hold 200 characters. If your program needs *more* characters than that—for example, if your program inputs 7 strings, each having 30 characters, so that the total number of characters is 210—the computer will gripe, and say "out of string space".

If you say—

```
1 CLEAR 1000
```

the computer will reserve enough space to hold 1000 unknown characters, instead of the normal quantity (which is just 50 or 200).

How much space can you reserve? That depends on how much memory you bought for your computer. If your computer has little memory, and you try to reserve more space than your computer has, the computer will give up, and say "out of memory".

How to turn a blank disk into a Radio Shack disk

If you buy a blank disk, you must turn it into a Radio Shack disk. Here's how. . . .

Extended color BASIC. Insert the blank disk into the disk drive; when you insert it, make sure the disk's label is on the *right* side of the disk (not the left), and make sure the disk's big oval cutout goes into the drive *before* the label does. Type DSKINI0. (That last character is a zero, not the letter O.) The disk drive's red light will glow for 40 seconds. When the light stops glowing, the computer will say OK.

Model 3 BASIC. Put the blank disk into the top disk drive. Type:

```
CMD "S"
BACKUP :0 :1
```

The computer will ask:

```
SOURCE Disk Master Password?
```

Type the word PASSWORD. The computer will copy all information from the bottom drive's disk to the blank disk. The blank disk will become a Radio Shack disk that's an exact copy of the other disk. The computer will say "TRSDOS Ready". Type the word BASIC. The computer will say "How Many Files". Press the ENTER key. The computer will say "Memory Size". Press the ENTER key. The computer will say READY.

Model 4 BASIC. Put the blank disk into the top drive. Type:

```
system
format :1 (q=n)
```

The computer will turn the blank disk into a Radio Shack disk. Then the computer will say "TRSDOS Ready". Type:

```
backup :0 :1
```

The computer will ask whether you're sure; type the letter y. The computer will copy all information from the bottom disk to the top disk. Then the computer will say "TRSDOS Ready". Type:

```
basic
```

The computer will say "Ready".

Radio Shack disk commands

The following commands all work, without any hassles: SAVE, LOAD, and RUN.

Here's how to make the computer print the names of all programs on the disk:

Extended color	Model 4	Model 3
DIR	system "dir"	CMD "D:0"

In extended color BASIC, you must say KILL "JOE.BAS" instead of KILL "JOE".

On any Radio Shack computer, your program's name can be short (such as JOE) or long: up to 8 characters. The name's first character must be a letter; the remaining characters should be letters or digits.

The main drive is called *drive 0*; the other drive is called *drive 1*. If you want JOE to be in drive 1 instead of drive 0, put a colon and a 1 after JOE, like this:

```
SAVE "JOE:1"
```

To make the computer print the names of all the programs in drive 1, type this:

Extended color	Model 4	Model 3
DIR 1	system "dir"	CMD "D:1"

TAPES

If you're using model 4 BASIC, you can't use tapes (and wouldn't want to anyway, since the model 4's disk system is far superior).

If you're using model 3 BASIC or extended color BASIC, use the following procedure to handle tapes. . . .

How to copy a program onto a tape

Position the tape and notice the counter's number. Press the recorder's RECORD and PLAY buttons, at the same time as each other. Type this:

```
CSAVE "A"
```

The computer will copy the program onto tape and then say OK or READY. Press the recorder's STOP button.

How to read a program from a tape

Put the tape into the recorder. (To do that, you might have to press the recorder's EJECT button.)

Type the word CLOAD.

Press the recorder's REWIND button. When the tape is rewound, press the recorder's STOP button. Push the counter's button, so the counter becomes zero.

Press the recorder's FAST FORWARD button. Look at the counter. When the number on the counter is almost up to the number of the program's beginning, press the recorder's STOP button.

Press the recorder's PLAY button. The computer will search on the tape for the beginning of the program. When the computer reaches the beginning of the program, extended color BASIC makes the computer print an F; model 3 BASIC makes the computer print two asterisks; the right asterisk flashes.

Then the computer begins reading the program. When the computer finishes reading it, the computer will say OK or READY.

Press the recorder's STOP button. Type the word LIST or RUN.

EDITING

To edit line 20, type:

```
EDIT 20
```

If the first character in line 20 is correct (and doesn't need to be edited), press the space bar. If the second character in line 20 is correct also, press the space bar again. Press the space bar for each correct character. Each time you press the space bar, the character that you said was correct appears on the screen.

When you get to the point where the next character on the screen would be *in*correct, press the D key instead of the space bar. The D key means: delete the next character. In extended color BASIC, pressing the D key prevents the next character from appearing on the screen; in model 3 BASIC and model 4 BASIC, pressing the D key causes the next character to be surrounded by funny marks that mean the character's being deleted.

To insert extra characters in the line, press the space bar several times, until you get to where you want the characters to be inserted. Then press the I key (which tells the computer you want to insert). Then type the characters you want to insert. After typing them, tap the up-arrow key while holding down the SHIFT key.

When you've finished all the deletions and insertions for that line, press the ENTER key. That makes the screen show the remainder of the line (the right-hand part of the line), and tells the computer you've finished editing the line.

RENUMBERING

Extended color BASIC and model 4 BASIC understand RENUM. In model 3 BASIC, say NAME instead of RENUM and you must be using disks.

PAUSING

To make the computer pause, hold down the SHIFT key; and while you keep holding down the SHIFT key, tap the @ key.

To stop pausing, tap any other key.

COUNTING TO 100

In model 3 and model 4 BASIC, to indent you can hit the space bar repeatedly; but to indent more easily, hit the right-arrow key once.

In extended color BASIC, the computer immediately forgets any indentation that you performed, so don't bother to indent.

ZONES

Each zone is 16 characters wide.

Model 4 BASIC produces 5 zones (so that the entire screen is 80 characters wide). Model 3 BASIC produces just 4 zones (so the screen's just 64 characters wide). Extended color BASIC produces just 2 zones (so the screen's just 32 characters wide).

TAB

In extended color BASIC and model 3 BASIC, the leftmost position in the line is called *position 0*; position 1 comes afterwards: so PRINT TAB(0) begins printing at the left margin, and PRINT TAB(1) begins printing afterwards.

LOCATE

Your computer doesn't understand the word LOCATE. Here's what to do instead . . .

Your computer considers the screen's top line to be *row 0*, and the line below it to be *row 1*. Your computer considers the row's leftmost character to be in *column 0*, and the next character to be in *column 1*.

To print the word DROWN, so that it begins at row 3 and column 7, type this:

Model 4: `print@(3,7), "drown"`
Model 3: `PRINT@64* 3+7, "DROWN"`
Extended color: `PRINT@32* 3+7, "DROWN"`

RANDOM NUMBERS

For a random decimal between 0 and 1, say RND(0) instead of RND.

In model 3 and model 4 BASIC, say RANDOM instead of RANDOMIZE. In extended color BASIC, omit the word RANDOMIZE altogether.

Model 4 BASIC. Unfortunately, model 4 BASIC can't handle pixels (unless you buy an expensive graphics board that nobody can afford).

Model 3 BASIC. In model 3 BASIC, you can use pixels. To make pixel (90,20) become white, say:

```
SET (90,20)
```

To make that pixel return to black, say:

```
RESET (90,20)
```

Whiten or blacken any pixel you wish: its X coordinate must be a number from 0 to 127; its Y coordinate must be a number from 0 to 47.

Extended color BASIC. In extended color BASIC, you can manipulate pixels in many ways. Let's see how.

The following explanation applies only to extended color BASIC. If you're using model 4 or model 3 BASIC instead, skip to the next section (which is entitled "MUSIC").

High-resolution graphics

To use high-resolution graphics, type these lines:

```
10 PMODE 4: PCLS: SCREEN 1
10000 GO TO 10000
```

Line 10 tells the computer to use high-resolution graphics: for each pixel, the X coordinate can go from 0 to 255, and the Y coordinate can go from 0 to 191. Line 10000, at the bottom of your program, makes the computer pause, so that you can admire whatever graphics have been drawn. Between lines 10 and 10000, you can insert lines containing the following commands. . . .

To draw a dot at pixel (100,100), say PSET (100,100).

To draw a line from pixel (0,0) to pixel (100,100), say LINE (0,0)-(100,100),PSET.

To draw a box ("rectangle") that has a corner at pixel (0,0) and has an opposite corner at pixel (100,100), say LINE (0,0)-(100,100),PSET,B.

To draw a box that has a corner at (0,0) and an opposite corner at (100,100), and also make the computer fill in the box (i.e., paint the box's interior), say LINE (0,0)-(100,100),PSET,BF.

Each of those commands contains the word PSET. To erase the shape that you drew, give the same command again, but replace the PSET by PRESET.

To draw a circle centered at pixel (100,100), and to make the circle's radius be 50, say CIRCLE (100,100),50. To erase that circle, say CIRCLE (100,100),50,0.

To paint the inside of any shape (such as a circle or a person), say PAINT and mention a point inside the shape. For example, to fill in an outline that has (101,101) inside it, say PAINT (101,101). To erase that paint (and also the shape itself), say PAINT (101,101),0.

To erase the entire drawing ("clear the screen"), say PCLS.

If you say SCREEN 0, the screen will stop showing graphics (and will show text instead), until you say SCREEN 1 (which brings back the graphics).

To get started experimenting with all those commands, run the following program; it draws a line from (0,0) to (100,100) and then draws a circle centered at pixel (100,100) and with radius 50:

```
10 PMODE 4: PCLS: SCREEN 1
20 LINE (0,0)-(100,100)
30 CIRCLE (100,100),50
10000 GO TO 10000
```

The computer draws the shapes in green, on a black background. After you've admired the pretty picture, abort the program, by pressing the BREAK key.

Modified high-resolution graphics

Modified high-resolution graphics are almost the same as high-resolution graphics. Here are the differences. . . .

To get modified high-resolution graphics, say PMODE 3 instead of PMODE 4.

In modified high-resolution graphics, whenever the computer draws (or erases) a pixel, it does the same thing to the pixel next to it. Specifically, if you tell the computer to draw (or erase) a pixel whose X coordinate is even, the computer will do the same thing to the pixel at its right (whose X coordinate is odd). If you tell the computer to draw (or erase) a pixel whose X coordinate is odd, the computer will do the same thing to the pixel at its left (whose X coordinate is even).

In modified high-resolution graphics, the computer normally draws red shapes on a green background. But if you don't like those colors, you can pick different colors instead.

To choose different colors, use these code numbers: 1=green, 2=yellow, 3=blue, and 4=red. For example, to draw yellow shapes on a blue background, say COLOR 2,3. The traditional place to say COLOR 2,3 is in line 10, like this:

```
10 PMODE 3: COLOR 2,3: PCLS: SCREEN 1
```

If you prefer, you can say COLOR 2,3 later in your program instead. If you say it later, the shapes at the beginning of your program will be unaffected by that COLOR statement. When you draw new shapes after giving that COLOR statement, the shapes that were already on the screen (in their old colors) remain unchanged; they keep their original colors.

When you say PRESET (in the middle of a dot or line or box command), the computer assumes you want the shape erased; to erase the shape, the computer redraws the shape by using the background color. If you recently changed the background color (by saying COLOR), the erasure will be done by using the new background color, which won't match the old background color.

If you say CIRCLE (100,100),50,2 the computer will draw the circle in yellow (because the 2 at the end of the CIRCLE command is the code number for yellow). Do *not* put a 0 at the end of the CIRCLE command, since 0 is not an acceptable color number. You must pick a number from 1 to 4.

If you say PAINT, you must say which pixel to start painting at; you must say which color paint to use; and you must remind the computer of what color the outline had that you're filling in. For example, if you want the computer to paint starting at pixel (101,101)—which is inside the outline— and you want the computer to use yellow paint (whose code number is 2), and the outline that you're filling in was drawn in blue (whose code number is 3), say PAINT (101,101),2,3.

Near the end of your program, if you say SCREEN 1,1, you'll see a "negative" of your picture. In the "negative", green becomes light blue (even though Radio Shack's manual incorrectly says it becomes "buff"), yellow becomes cyan (greenish blue), blue becomes magenta (purplish red), and red becomes orange. All future drawings (even by other programs) will be done using those "negative" colors, unless you tell the computer to return to a "positive", by saying SCREEN 1,0.

MUSIC

Model 3 BASIC can't play music. Model 4 BASIC plays music so poorly that I won't bother to discuss it. So if you have model 3 BASIC or model 4 BASIC, skip ahead to the next section (entitled "EXPONENTS").

Extended color BASIC plays music wonderfully. Here are the details. . . .

How to SOUND

If you type—

```
SOUND 100,3*15
```

the computer will produce a sound. In that command, the 100 is the pitch: you can choose any pitch from 1 to 255. The 3 is how many seconds the sound will last: you can choose any number of seconds from .07 to 17. The *15 is needed, because the computer's metronome ticks 15 times per second.

How to PLAY

You can type this:

```
PLAY "CDG#B-A"
```

The computer will play the note C, then D, then G sharp, then B flat, then A.

The computer can play in five octaves, numbered from 1 to 5. Octave 1 consists of very bass notes; octave 5 consists of very high-pitched notes. In each octave, the lowest note is a C: the notes in an octave are C, C#, D, D#, E, F, F#, G, G#, A, A#, and B.

"Middle C" is at the beginning of octave 2. Normally, the computer plays in octave 2. To make the computer switch to octave 4, type the letter "O" followed by a 4, like this:

```
PLAY "O4"
```

After giving that command, anything else you PLAY will be in After giving that command, anything else you PLAY will be in octave 4, until you change octaves again.

Besides playing with pitches, you can also play with rhythms ("lengths" of the notes). Normally each note is a "quarter note". To make the computer switch to eighth notes (which are faster), type this:

```
PLAY "L8"
```

Besides using L8 for eighth notes, you can use L16 for sixteenth notes (which are even faster) and L32 for thirty-second notes (which are super-fast). For long notes, you can use L2 (which gives a half note) or L1 (which gives a whole note).

In fact, you can use any length from L1 to L255. You can even use in-between lengths, such as L7 or L23 (though such rhythms are hard to stamp your foot to).

If you put a period after the length, the computer will multiply the length by 1½. For example, if you say "L8." instead of just "L8", the computer will play the notes 1½ times as long as 8th notes. Musicians call them "dotted eighth notes."

To make the computer pause ("rest") for an eighth note, put a P8 into the music string.

Normally, the computer plays at tempo 2 (which is 112 quarter notes per minute); but you can change the tempo. To switch to tempo 6 (which is three times as fast as tempo 2), say:

```
PLAY "T6"
```

You can switch to any tempo from 1 to 255. The 1 is very slow; 255 is ridiculously fast. In musical terms, 1 is largo, 2 is moderato, 3 is vivace, and 4 (or more) is even faster than prestissimo!

Normally, the computer plays at volume 15; but you can change the volume. To switch to volume 4 (which is lower, quieter), say:

```
PLAY "V4"
```

You can switch to any volume from 1 to 31.

You can combine all those musical commands into a single PLAY statement. For example, to set the tempo to 6, the volume to 4, the octave to 4, the length to 8 (which means an eighth note), and then play C and D, and then change the length to 4 and play E, type this:

```
PLAY "T6V404L8CDL4E"
```

EXPONENTS

For exponents, extended color BASIC uses the symbol "↑"; model 3 BASIC uses the symbol "[" (which you type by pressing the up-arrow key); model 4 BASIC uses the symbol "^" (which you type by pressing the semicolon key while holding down the CLEAR key).

CP/M COMPUTERS

SCOPE

The CP/M operating system is used by many computers. The most popular computer for using CP/M is the Kaypro, because it costs the least (prices start at $1595). Other computers that use CP/M are manufactured by Morrow, Cromemco, Vector, North Star, Xerox, and Zorba. Another company that made CP/M computers was Osborne; although Osborne recently went bankrupt, many homes and offices bought Osborne computers and still use them.

All those computers were designed especially for running CP/M. Their prices include CP/M at no extra charge.

On other popular brands (such as Apple, IBM, Radio Shack, and Epson), CP/M is an extra-cost option.

The DEC Rainbow runs CP/M-86, which is very similar to CP/M.

On all those computers, you can use several versions of BASIC, such as MBASIC, SBASIC, CBASIC, and CB-80.

Begin by learning *MBASIC* (which is also called *Microsoft BASIC* and *BASIC-80*), since that's the version of BASIC that's easiest to learn. It was invented by Bill Gates, who heads Microsoft. It's excellent. If you use MBASIC, you can run the examples in The Secret Guide to Computers without any problem.

Here's how to use MBASIC. I assume you have the most recent version of MBASIC: it was invented several years ago and is called version 5. (Do *not* use version 4, which is older and obsolete.)

The following explanation of MBASIC (version 5) applies to all CP/M computers. Since the most popular CP/M computer is the Kaypro, I'll make some extra comments (in parentheses) about the peculiarities of the Kaypro.

THE KEYBOARD

On most CP/M computers (such as the Kaypro), the zero key is Swedish.

On most CP/M computers (such as the Kaypro), the letters are *not* automatically capitalized. If you press the R key, you'll get a small r. To get a capital R, you must hold down the SHIFT key.

GET STARTED

To get started, you need a disk that contains both MBASIC and CP/M.

(When you buy a Kaypro, you get MBASIC on one disk and CP/M on a separate disk. To combine those two disks into a single disk, you must give a SYSGEN command, which is explained in the manual that came with your Kaypro.)

If your computer has just one disk drive, that drive is called *drive A*. If your computer has two disk drives, the drive that's on the top or left is called *drive A*; the other drive is called *drive B*.

(For example, the Kaypro's top drive is called *drive A*; its bottom drive is called *drive B*.)

To start, put the disk that contains both MBASIC and CP/M into drive A. Then turn on the computer's power. (The Kaypro's power switch is at the Kaypro's rear.)

The computer will say:

```
A>
```

Type the word "mbasic", like this:

```
A>mbasic
```

Then press the RETURN key. The computer will say:

```
Ok
```

That procedure works on the Kaypro, but some other computers are different. Instead of "mbasic", some computers require you to type "basic"; and some computers don't require you to type anything at all!

ARITHMETIC

To type the word PRINT, you can use either capital letters or small letters; the computer doesn't care.

GRIPES

If the computer gripes at you, press the RETURN key immediately.

After pressing the RETURN key, correct the error, by retyping the line that the computer griped about.

HOW TO PRINT ON PAPER

To print on paper, say LPRINT instead of PRINT. For example, to make the computer say "I love you" on paper, type this:

```
lprint "I love you"
```

REARRANGING YOUR PROGRAM

To LIST your program on paper instead of on your screen, type LLIST instead of LIST.

CLEARING THE SCREEN

The CLS command doesn't work. The method of clearing the screen depends on which computer you have: read the manual that came with your computer.

GO TO

Instead of pressing a BREAK key, do the following: hold down the CONTROL key (which on the Kaypro is marked "CTRL"); and while you keep holding down the CONTROL key, tap the C key.

NUMERIC VARIABLES

A numeric variable's name can be a letter (such as X) or a longer combination of characters (such as profit.in.1984.before.November.promotion). For example, you can type:

```
10 profit.in.1984.before.November.promotion = 3497.18
20 profit.in.1984 = profit.in.1984.before.November.promotion + 6214.27
30 print profit.in.1984
```

The computer will print:

```
 9711.45
```

The variable's name can be quite long: up to 40 characters! The first character in the name must be a letter; the remaining characters can be letters, digits, or periods. The name must not be a word that has a special meaning to the computer; for example, the name cannot be PRINT or GO or TO.

STRING VARIABLES

A string variable looks like a numeric variable, but has a dollar sign after it. For example, you can have a string variable such as:
my.job.in.1984.before.November.promotion$

DISKS

How to turn a blank disk into a CP/M MBASIC disk

If you buy a blank disk and want your computer to use it, you should turn it into a CP/M MBASIC disk. Here's how to do it, if you have a Kaypro 2 or Kaypro 4. (If you have a Kaypro 10 or a different brand of computer, read the manual that came with your computer.)

For a Kaypro 2 or Kaypro 4, put the blank disk into drive B. Then type the following:

```
system
copy
c
```

After that final c, press the RETURN key *twice*. The computer will copy all information from drive A's disk to the blank disk. The blank disk will become a CP/M MBASIC disk that's an exact copy of the other disk. After the copying is complete, return to BASIC by typing:

```
mbasic
```

Disk commands

To make the computer copy your program from the RAM to the disk, and name the program JOE, type this:

```
save "JOE"
```

The program's name can be short (such as JOE) or long: up to 8 characters. In the program's name, you must capitalize every letter. The name's first character must be a capital letter; the remaining characters should be capital letters or digits.

To erase JOE from the disk, you can*not* say KILL "JOE". Instead, you must say:

```
kill "JOE.BAS"
```

The "BAS" emphasizes that JOE was written in BASIC.

If you want JOE to be in drive B instead of drive A, put a B and a colon before JOE, like this:

```
save "B:JOE"
```

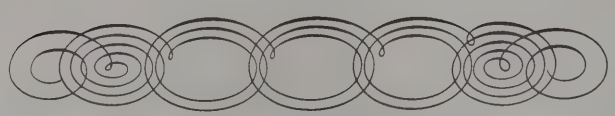
Remember to capitalize the B and the JOE.

To make the computer print the names of all the programs in drive B, type this:

```
files "B:*.*"
```

TAPES

Hardly anybody who buys CP/M uses tapes. Therefore, I won't bother to explain CP/M's tape system.



EDITING

To edit line 20, type:

```
edit 20
```

If the first character in line 20 is correct (and doesn't need to be edited), press the space bar. If the second character in line 20 is correct also, press the space bar again. Press the space bar for each correct character. Each time you press the space bar, the character that you said was correct appears on the screen.

When you get to the point where the next character on the screen would be *in*correct, press the D key instead of the space bar. The D key means: delete the next character. When you press the D key, the next character will be surrounded by backslashes that mean the character's being deleted.

To insert extra characters in the line, press the space bar several times, until you get to where you want the characters to be inserted. Then press the I key (which tells the computer you want to insert). Then type the characters you want to insert. After typing them, press the key that's marked ESCAPE (or ESC).

When you've finished all the deletions and insertions for that line, press the RETURN key. That makes the screen show the remainder of the line (the right-hand part of the line), and tells the computer you've finished editing the line.

PAUSING

To make the computer pause, hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the C key.

To stop pausing, tap any other key.

ZONES

On a typical CP/M computer (such as a Kaypro), the monitor prints 80 characters per line, and the screen has five zones: zones 1-4 are each 14 characters wide; zone 5 is 24 characters wide.

LOCATE

Your computer doesn't understand the word LOCATE. To find out what to type instead of LOCATE, read the manual that came with your computer.

RANDOM NUMBERS

Omit the RANDOMIZE command.

For a random integer between 1 and 5, say 1+INT(RND*5) instead of RND(5).

PIXELS

Most CP/M computers can't handle pixels.

MUSIC

Most CP/M computers can't handle music.

EXPONENTS

For exponents, most CP/M computers use the symbol "^" instead of "↑".

DEC

Digital Equipment Corporation, which is nicknamed "DEC", makes several computers. DEC's most popular computers are the Rainbow (which is a micro), the PDP-11 (which is a mini), the VAX-11 (which is a larger mini), the Decsystem 10 (which is a maxi), and the Decsystem 20 (which is a newer version of the Decsystem 10).

The Rainbow is a micro that resembles the IBM PC. It can handle all the important programs in The Secret Guide to Computers.

DEC's larger computers (the PDP-11, VAX-11, Decsystem 10, and Decsystem 20) use DEC's own versions of BASIC. DEC's most popular versions are called *BASIC-PLUS* and *BASIC-PLUS-2*, which are quite similar to each other. Here are their peculiarities. . . .

The keyboard. The keyboard depends on which terminal you're using. For details, read the manual that came with your computer and also the manual that came with your terminal.

Get started. The way to get started depends on which version of BASIC you're using; for details, read the manual that came with your computer. When you've finished getting started, the computer will say READY.

Programming. Above the program, say NEW JOE instead of just NEW. If you're using BASIC-PLUS (instead of BASIC-PLUS-2), your program's bottom line should say END, like this:

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
3 PRINT "LET'S GET MARRIED"
4 END
```

How to print on paper. Use a terminal that has paper coming out of it, instead of a screen.

Clearing the screen. Your computer does not understand CLS. To find out how to clear the screen, read the manuals that came with your computer and terminal.

GO TO. To abort a program, hold down the CONTROL key (which is marked CONTROL or CTRL); and while you keep holding down that key, tap the C key twice.

Numeric variables. If you're using BASIC-PLUS, a numeric variable's name can be a letter (such as X) or a letter-followed-by-a-digit (such as X2). If you're using BASIC-PLUS-2, a numeric variable's name must start with a letter and can be a combination of up to 30 letters, digits, and periods, such as:
PROFIT.IN.1984.BEFORE.NOVEMBER

String variables. A string variable looks like a numeric variable, but has a dollar sign after it.

Colons. In BASIC-PLUS, the colon works. In BASIC-PLUS-2, type a backslash (\) instead of a colon (:).

END. The only place in your program where you can say END is the bottom line. No other line in your program can say END. If you want to make the computer to stop before it reaches the bottom of your program, insert a line in your program that says STOP.

Disks. Do *not* put quotation marks around the program's name. Instead of saying FILES, say CATALOG; instead of saying LOAD, say OLD; instead of saying KILL, say UNSAVE. Instead of saying RUN "JOE", say:

```
OLD JOE
RUN
```

If JOE is already on the disk, and you want to replace it by a newer version of JOE, you must say REPLACE JOE instead of SAVE JOE.

Tapes. Hardly anybody uses tape on DEC's computers, so I won't bother to discuss tape.

Editing. DEC's BASIC doesn't provide any easy way of editing a previous line. You must type the whole line over again.

Renumbering. Instead of saying RENUM, say RES (which stands for the word RESEQUENCE).

Pausing. To pause, hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the S key. To stop pausing, hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the Q key.

Apostrophe. Instead of an apostrophe, type an exclamation point.

Lists of data. If you're using BASIC-PLUS (instead of BASIC-PLUS-2), you must put quotation marks around each DATA string containing lower-case letters or blank spaces.

Counting to 100. After the word NEXT, you must put a variable: you can say "NEXT X", but you can*not* say just "NEXT".

Zones. Each zone is 14 characters wide, except the rightmost zone (which is wider).

TAB. The leftmost position in the line is called *position 0*; position 1 comes afterwards: so PRINT TAB(0) begins printing at the left margin, and PRINT TAB(1) begins printing afterwards. If you're using BASIC-PLUS-2 (instead of BASIC-PLUS), you must put a semicolon after the TAB's right parenthesis.

LOCATE. Your computer doesn't understand the word LOCATE.

Random numbers. For a random integer between 1 and 5, say 1+INT(RND*5) instead of RND(5).

Pixels. Your computer can't handle pixels.

Music. Your computer can't handle music.

Exponents. For exponents, most DEC terminals use the symbol "^" instead of "↑".

Several kinds of minicomputers and microcomputers have been manufactured by Texas Instruments (which is nicknamed "TI").

TI's minicomputers are unpopular, so I won't discuss them. On the other hand, TI's *micro*computers have been *very* popular. TI's main microcomputers have been the *TI Professional Computer* and the *TI 99/4A Home Computer*.

The TI Professional Computer, which costs about $2500, resembles the IBM PC; so if you have a TI Professional Computer, read the section of The Secret Guide that describes the IBM PC. The main difference between the TI Professional Computer and the IBM PC is that the keys are labeled slightly differently. For details about TI's keyboard, read the manual that TI gave you.

The TI 99/4A Home Computer, which TI has stopped manufacturing, has many peculiarities; let's look at them. . . .

The keyboard. The quotation mark is on the *front* of the P key: to type the quotation mark, tap the P key while holding down the FCTN key. Similarly, to type a question mark (which is on the front of the I key), tap the I key while holding down the FCTN key. To backspace, type a left-arrow (which is on the front of the S key and therefore requires the FCTN key). The RETURN key says ENTER on it.

Get started. Turn on the computer's power switch. (It's on the front of the computer, near the front right corner). Turn on the TV or monitor. If you're using a TV, turn the TV to channel 3 or 4. You'll see the words TEXAS INSTRUMENTS on the screen. Press any key. Another screenful of information will appear. If you own TI's Extended BASIC cartridge and inserted it, type a 2; otherwise, type a 1. The computer will print the symbol ">".

How to print on paper. Since hardly anybody buys the TI 99/4A's printer, I won't bother to discuss it.

Rearranging your program. Your computer doesn't understand the word DELETE.

Clearing the screen. To clear the screen, say CALL CLEAR instead of CLS.

GO TO. To abort a program, type the number 4 while holding down the FCTN key.

Numeric variables. A numeric variable's name can be a letter (such as X) or a longer combination of characters (such as PROFIT_IN_1984). The variable's name can be long: up to 15 characters. The first character in the name must be a letter; the remaining characters can be letters, digits, or underlines.

String variables. A string variable looks like a numeric variable, but has a dollar sign after it, and has no more than 15 characters total (including the dollar sign).

String input. In the INPUT statement, use a colon instead of a semicolon; and at the end of the prompt, put a question mark followed by a space followed by a quotation mark. . . .

Most computers: `10 INPUT "WHAT IS YOUR NAME";N$`
TI 99/4A computer: `10 INPUT "WHAT IS YOUR NAME? ":N$`

Colons. On most computers, if you want to put two statements on the same line, you put a colon between the statements. On the TI 99/4A computer, use a *pair* of colons, like this:

```
10 A=5:: B=7
```

Moreover, your computer understands the pair of colons only if you buy the *TI Extended BASIC cartridge*, which costs about $90 extra!

IF . . . THEN. To use IF . . . THEN statements easily, you must buy the TI Extended BASIC cartridge. (Without that cartridge, the TI won't let you say "AND", won't let you say "OR", and won't let you put statements after the word THEN: instead of statements, you must put the number of a line where you want the computer to go.) Since most programs in The Secret Guide to Computers contain IF . . . THEN statements, you'll have difficulty using the Guide unless you buy that cartridge.

Disks. Since hardly anybody buys disks for the TI 99/4A, I won't bother to discuss them.

Tapes. To copy your program onto a tape, type SAVE CS1 (which means "save on cassette recorder #1"). After you type that, the computer will chat with you and explain how to handle your recorder. When it asks you to type Y or N, make sure you type a large capital Y or N (by holding down the SHIFT key). To copy the program from the tape back to the computer, say OLD CS1; the computer will chat with you; after the computer stops chatting, type LIST.

Editing. To edit line 20, type the number 20 and then type an down-arrow (by holding down the FCTN key). The computer will show you line 20. Move to the part of line 20 that you want to change, by using the left-arrow and right-arrow keys (which require the FCTN key). To delete a character, type the number 1 while holding down the FCTN key. To insert a character, type the number 2 while holding down the FCTN key; that makes the computer create an extra space, to put the character into. After editing (by using the left-arrow, right-arrow, FCTN 1, and FCTN 2, and typing your new characters), you can press the ENTER key. If you want to edit the next line also, type a down-arrow instead of pressing the ENTER key. To type a down-arrow, remember to hold down the FCTN key.

Renumbering. Instead of typing RENUM, type RES (which stands for RESEQUENCE).

Pausing. To make the computer pause, give an abortion; then make the computer continue (by typing CON).

Apostrophe. Instead of an apostrophe, type an exclamation point. Moreover, the exclamation point works only if you have the TI Extended BASIC cartridge.

Counting to 100. After the word NEXT, you must put a variable: you can say "NEXT X", but you can*not* say just "NEXT".

Zones. Your computer has just 2 zones. Each zone is 14 characters wide, so the entire screen is 28 characters wide.

TAB. After the TAB's right parenthesis, you must put a semicolon.

LOCATE. Your computer doesn't understand the word LOCATE. To print the word DROWN, so that it begins at row 3 and column 7, type this:

```
DISPLAY AT (3,7): "DROWN"
```

That works only if you have the TI Extended BASIC cartridge.

Random numbers. For a random integer between 1 and 5, say 1+INT(RND*5) instead of RND(5).

Pixels. Your computer can't easily handle pixels.

Sound. If you type—

```
CALL SOUND(2000,440,0)
```

the computer will make a sound for 2000 milliseconds (which is 2 seconds), at a frequency of 440 cycles per second (which is "the A below middle C"), unmuted (muting=0). The sound will come from your TV or monitor; make sure you've turned up its volume dial. You can choose any duration from 1 to 4250 milliseconds, any frequency from 110 to 44733, and any muting from 0 to 30. For a chord made of four such sounds, type this:

```
CALL SOUND (2000,440,0,500,0,600,0,700,0)
```

Exponents. For exponents, use the symbol "^" instead of "↑".

TIMEX SINCLAIR

Sinclair is a British company that's invented several computers: the ZX-80, the ZX-81, and the ZX Spectrum. In the United States, Timex sells slightly improved versions of those computers: the Timex Sinclair 1000 computer is a slightly improved ZX-81; the Timex Sinclair 1500 is a further improved ZX-81; and the Timex Sinclair 2068 is an improved ZX Spectrum.

The ZX-80, ZX-81, Timex Sinclair 1000, and Timex Sinclair 1500 are all inadequate. For example, they don't understand the words DATA and READ, they don't understand colons, and they have trouble handling the word IF. Therefore, I recommend that you avoid those computers.

On the other hand, the ZX Spectrum and Timex Sinclair 2068 are fine; here's how to use them. . . .

The keyboard. Instead of typing the word PRINT, press the key that says PRINT on it. Instead of typing the word NEW, press the key that says NEW on it. Each computer word has its own key, so never type a computer word: press the computer word's key instead.

The key that says NEW also says STOP. If you press that key, you'll be typing NEW. If you press that key *while holding down the SYMBOL SHIFT key*, you'll be typing STOP instead.

That's why STOP is written on the key in a strange color. (On the ZX Spectrum, the word STOP is written in red; on the Timex Sinclair 2068, the word STOP is written in silver on a black background.) The strange color means: you must hold down the SYMBOL SHIFT key.

Notice that the math symbols (plus, minus, times, and divide) and quotation mark are all written in a strange color. So to type those symbols, you must hold down the SYMBOL SHIFT key.

To backspace, press the DELETE key *while holding down the CAPS SHIFT key*.

The RETURN key says ENTER on it.

Gripes. If the computer gripes at you by printing a question mark in the middle of a line, backspace until the question mark goes away.

Spaces. The computer does *not* automatically print a blank space after each number, and does *not* automatically print a blank space before each number that's positive.

Numeric variables. Instead of saying "X=47", you must say "LET X=47".

END. Say STOP instead of END.

Tapes. To copy your program onto tape, say SAVE. To copy your program from tape to the computer, say LOAD; then LIST the program.

Counting to 100. After the word NEXT, you must put a variable: you can say "NEXT X", but you can*not* say just "NEXT".

ATARI

Atari offers two versions of BASIC. The old version (which is crummy) is called just *Atari BASIC*. The new version (which is excellent and costs $90) is called *Atari Microsoft BASIC version 2*. To run the programs in The Secret Guide to Computers easily, you must buy Atari Microsoft BASIC version 2.

If you *don't* buy Atari Microsoft BASIC version 2—if you try using the old Atari BASIC instead—you face five difficulties:

1. For each string variable, you must say DIM. For example, if you want G$ to be "DOWN", you must say:

```
5 DIM G$(4)
10 G$="DOWN"
```

The DIM G$(4) warns the computer that G$ will stand for a string that has 4 characters.

2. The computer does *not* automatically print a blank space after each number, and it does *not* automatically print a blank space before each number that's positive.

3. The INPUT statement cannot contain a prompt. Instead of saying—

```
10 INPUT "WHAT IS YOUR NAME";N$
```

you must say:

```
10 PRINT "WHAT IS YOUR NAME";
11 INPUT N$
```

In that example, since N$ is a string variable, you must DIM it, by adding this line:

```
5 DIM N$(15)
```

That tells the computer to permit N$ to be a string of up to 15 characters.

4. After the word "NEXT", you must put a variable: you can say "NEXT X", but you can*not* say just "NEXT".

5. If a FOR . . . NEXT loop contains a GO TO statement that makes the computer skip out of the loop, you should say "POP: GO TO" instead of "GO TO". If the GO TO statement makes the computer skip out of *two* loops at once, you should say "POP: POP: GO TO" instead of "GO TO".

APPENDIX 2

LISTS

ESSENTIALS

MANUFACTURERS

Amdahl 18
Apple 19, 101, 288
Atari 26, 99

Burroughs 18

Cambex 18
Coleco (Adam) 19, 102
Columbia 18
Commodore 19, 97, 297
Compaq 18
Control Data Corp. (CDC) 18
Corona 18
Cromemco 20, 103

Data General 18
Digital Eq. Corp. (DEC) 18, 106

Eagle 18
Epson 20, 104

Franklin (Ace) 19, 26, 102

Hewlett-Packard (HP) 18
Honeywell 18

Inter. Bus. Mach. (IBM) 18, 104, 306
IPL 18

Kaypro 20, 26, 103

Micro Professor 19
Morrow 20, 103

National Adv. Sys. (NAS) 18
NCR 18
Nixdorf 18
North Star 20

Orange 19
Osborne 20, 26

Pear 19
Pineapple 19
Prime 18

Seequa (Chameleon) 18
Sperry (Univac) 18

Tandy (Radio Shack) 18, 100, 317
Texas Instruments (TI) 26, 98
Timex (Timex/Sinclair) 26, 98

Vector 20

Wang 18

Xerox 20

Zorba 20

HARDWARE

Centronics int. 94
CPU 78
disk 24, 84, 110
disk drive 24, 84, 110
keyboard 21, 35
LCD 25
modem 95
monitor 24
printer 27, 45, 88
printer cable 94
RAM 23, 79
RCA cord 21
ROM 28, 81
RS-232 interface 95
switch box 22
tape 23, 82, 112
tape drive 82
tape recorder 23, 82, 112
TV 21

OPERATING SYSTEMS

Apple DOS 29
CP/M 20, 29, 103
CP/M-86 104, 107
MS-DOS 29
TRSDOS 29
Valdocs 104

LANGUAGES

BASIC 28, 34
COBOL 28
FORTRAN 28
LOGO 283
PASCAL 28

APPLICATIONS

accounting 30, 101, 158
art 212
artificial intell. 159, 250, 279
astrology 166
bartending 270
calculus 281
data management 30, 158
education 101, 159, 271
electronic mail 156
games 154, 164, 254
literary analysis 246, 248, 262
medicine 165, 270, 272
music 155, 280
poetry 234
psychology 166, 279
robots 155, 276, 283
sex 160, 242
vision 282
word processing 30, 156

MODERN MICROSOFT BASIC

```
CLS                              52
DATA MEAT,POTATOES              116
DELETE 30-80                     49
DIM X(8)                        176
END                              68
FILES                           110
FOR X = 1 TO 100                122
GO TO 10                         53
GOSUB 1000                      184
IF A$="MALE" THEN PRINT          68
INPUT "WHAT NAME";N$             62
KILL "JOE"                      111
LIST                             49
LOAD "JOE"                      111
LOCATE 3,7                      133
NEW                              42
NEXT X                          122
PRINT 5+2                        37
RANDOMIZE                       136
READ A$                         116
RENUM                           114
RESTORE                         117
RETURN                          184
RUN                              42
SAVE "JOE"                      110
X=47                             58
? 5+2                            38
'THIS PROGRAM IS DUMB           115
```

COMMODORE EXTRAS

```
CATALOG DO                      301
CLOSE 4                         298
CLOSE 15                        300
CMD 4                           298
CONT                            303
DLOAD "JOE"                     301
DSAVE "JOE"                     301
HEADER "ELIZABETH",I00          300
OPEN 4,4                        298
OPEN 15,8,15,"UI-"              300
POKE V+5,4                      305
PRINT#4                         298
PRINT#15, "I"                   301
REM BECAUSE RUSS HAS 30         303
SCRATCH "JOE"                   301
```

IBM EXTRAS

```
BASICA                          306
BEEP                            314
CIRCLE (100,100),50             312
COLOR 1                         313
DISKCOPY A: B:                  309
EDIT 20                         310
LINE (0,0)-(100,100)            312
LLIST                           307
LPRINT "I LOVE YOU"             307
PAINT (100,101)                 312
PLAY "CDG#B-A"                  315
PSET (100,100)                  312
SCREEN 1                        312
SOUND 440,18.2                  314
SYSTEM                          309
WIDTH 40                        307
```

APPLE EXTRAS

```
CATALOG                         291
COLOR=11                        294
DEL 30,80                       289
DELETE JOE                      291
GR                              294
HCOLOR=1                        295
HGR                             295
HLIN 4,8 AT 19                  294
HOME                            290
HPLOT 250,140                   295
HTAB 7                          293
INIT HELLO                      291
PLOT 31,37                      294
PR#0                            289
PR#1                            289
REM BECAUSE RUSS HAS 30         293
SPEED=0                         290
TEXT                            294
VLIN 20,30 AT 4                 294
TAB 3                           293
&                               293
```

RADIO SHACK EXTRAS

```
BACKUP :0 :1                    321
BASIC                           317
CIRCLE (100,100),50             324
CLEAR 1000                      320
CLOAD                           322
CMD "D:0"                       321
CMD "S"                         321
CMD "Z","ON"                    318
CSAVE "A"                       322
DEL 30-80                       319
DIR                             321
EDIT 20                         322
FORMAT :1 (Q=N)                 321
LINE (0,0)-(10,10),PSET         324
LLIST                           319
LPRINT "I LOVE YOU"             318
PAINT (101,101)                 324
PCLS                            324
PLAY "CDG#B-A"                  326
PMODE 4                         324
PRESET (100,100)                324
PRINT#-2, "I LOVE YOU"          318
PRINT@(3,7), "DROWN"            323
RANDOM                          323
RESET (90,20)                   324
SCREEN 1                        324
SET (90,20)                     324
SOUND 100,3*15                  326
SYSTEM "DIR"                    321
SYSTEM "LINK *DO *PR"           318
SYSTEM "RESET *DO"              318
```

CP/M EXTRAS

```
COPY                            330
EDIT 20                         331
LLIST                           329
LPRINT                          329
MBASIC                          328
SYSTEM                          330
```

MODELS

ARITHMETIC

```
PRINT 5+2
```

The computer will print 7. (page 37)

QUESTION MARK

```
? 5+2
```

The computer will print 7. (page 38)

ORDER OF OPERATIONS

```
PRINT 2+3*4
```

The computer will print 14. (page 39)

PARENTHESES

```
PRINT 5-(1+1)
```

The computer will print 3. (page 39)

STRINGS

```
PRINT "I LOVE YOU"
```

The computer will print I LOVE YOU. (page 40)

PROGRAMMING

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
3 PRINT "LET'S GET MARRIED"
```

If you type the word NEW above the program, and the word RUN below the program, the computer will print the three strings on separate lines. (page 42)

SEMICOLONS

```
10 PRINT "FAT";
20 PRINT "HER"
```

The computer will print FATHER. (page 50)

BLANK LINES

```
10 PRINT "JOY"
20 PRINT
30 PRINT "SORROW"
```

The computer will print JOY, then a blank line, then SORROW. (page 50)

SPACES NEAR NUMBERS

```
PRINT 7;"DO";"NUTS"
```

The computer will print 7 DONUTS. It will put a blank space before and after the 7. (page 51)

CLEARING THE SCREEN

```
10 CLS
20 PRINT "I'M SITTING ON TOP"
```

The computer will clear the screen, then print I'M SITTING ON TOP. (page 52)

GOING BACK

```
10 PRINT "CAT"
20 PRINT "DOG"
30 GO TO 10
```

The computer will print CAT and DOG repeatedly. (page 53)

SKIPPING AHEAD

```
10 PRINT "YOUR MOTHER"
15 GO TO 30
20 PRINT "HATES ANYONE WHO"
30 PRINT "KISSES DEAD SNAKES"
```

The computer will print YOUR MOTHER and KISSES DEAD SNAKES. (page 57)

NUMERIC VARIABLES

```
10 X=47
20 PRINT X+2
```

The computer will print 49. (page 58)

STRING VARIABLES

```
10 G$="DOWN"
20 PRINT G$
```

The computer will print DOWN. (page 39)

INPUT

```
10 INPUT "WHAT IS YOUR NAME";N$
20 PRINT "I HATE THE NAME ";N$
```

The computer will ask WHAT IS YOUR NAME? If you answer JOEY, the computer will say I HATE THE NAME JOEY. (page 62)

COLONS

```
10 A=5: B=7: PRINT A+B
```

The computer will print 12. (page 68)

END

```
10 PRINT "BUBBLE GUM"
20 END
30 PRINT "FOX"
```

The computer will print BUBBLE GUM. (page 68)

IF . . . THEN

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "TOO BAD"
```

The computer will ask ARE YOU MALE OR FEMALE? If you answer MALE, the computer will say TOO BAD. (page 68)

APOSTROPHE

```
10 'THIS PROGRAM IS A DUMB EXAMPLE
20 C=30' BECAUSE RUSS HAS 30 COMPUTERS
30 PRINT C
```

The computer will print 30. (page 115)

DATA . . . READ

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES
20 READ A$
30 PRINT A$
40 GO TO 20
```

The computer will print MEAT and POTATOES and LETTUCE and TOMATOES and then say OUT OF DATA. (page 116)

DATA END

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES
15 DATA END
20 READ A$: IF A$="END" THEN END
30 PRINT A$
40 GO TO 20
```

The computer will print MEAT and POTATOES and LETTUCE and TOMATOES. (page 117)

RESTORE

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES
15 DATA END
20 READ A$
21 IF A$="END" THEN RESTORE: GO TO 20
30 PRINT A$
40 GO TO 20
```

The computer will print MEAT and POTATOES and LETTUCE and TOMATOES, repeatedly. (page 117)

FOR . . . NEXT

```
10 FOR X = 1 TO 100
20     PRINT X
30 NEXT X
```

The computer will print every number from 1 to 100. (page 122)

STEP

```
10 FOR I = 10 TO 1 STEP -1
20     PRINT I
30 NEXT
```

The computer will print every number from 10 down to 1. (page 127)

ZONES

```
PRINT "SIN","KING"
```

The computer will print SIN in the first zone and print KING in the second zone. (page 128)

TAB

```
PRINT TAB(6)"HOT"
```

The computer will skip to position 6 and then print HOT. (page 130)

LOCATE

```
LOCATE 3,7: PRINT "DROWN"
```

The computer will print the word DROWN, beginning at line 3's 7th position. (page 133)

RANDOM NUMBERS

```
10 RANDOMIZE
20 PRINT RND(5)
```

The computer will print 1 or 2 or 3 or 4 or 5. (page 136)

RANDOM DECIMALS

```
10 RANDOMIZE
20 PRINT RND
```

The computer will print a random decimal between 0 and 1. (page 140)

INCREMENTING

```
10 A=5
20 A=3+A
30 PRINT A
```

The computer will print 8. (page 141)

SIMPLE SUBSCRIPTS

```
10 DIM X(3)
20 X(1)=57.2
30 X(2)=-8.3
40 X(3)=476
50 FOR I = 1 TO 3: PRINT X(I): NEXT
```

The computer will print 57.2 and -8.3 and 476. (page 176)

DOUBLE SUBSCRIPTS

```
10 DIM X(2,3)
20 X(1,1)=57
30 PRINT X(1,1)
```

The computer will print 57. (page 181)

STRING ARRAYS

```
10 DIM X$(3,3)
20 X$(1,1)="DOG"
30 PRINT X$(1,1)
```

The computer will print DOG. (page 183)

SUBROUTINES

```
10 PRINT "THE"
20 GOSUB 1000
30 PRINT "DEAD"
40 END
1000 PRINT "BIRD"
1010 PRINT "IS"
1200 RETURN
```

The computer will print THE, BIRD, IS, DEAD. (page 184)

SQUARE ROOTS

```
PRINT SQR(9)
```

The computer will print 3. (page 193)

ABSOLUTE VALUE

```
PRINT ABS(-7)
```

The computer will print 7. (page 193)

ROUNDING

```
PRINT INT(17.9)
```

The computer will print 17. (page 194)

ERROR MESSAGES

If you type a command that the computer can't obey, the computer will gripe, by printing an *error message*.

Here's a list of the 25 most popular error messages, in the order you'll probably encounter them.

I'll show you the error messages printed by the IBM Personal Computer. On other popular computers (by Apple, Commodore, Radio Shack, the CP/M cartel, etc.), the error messages are worded slightly differently but are almost the same. In parentheses, I'll show you the two-letter abbreviation used by the Radio Shack Color Computer.

If you say PRIND instead of PRINT, the computer will say SYNTAX ERROR (or SN ERROR). That means the computer hasn't the faintest idea of what you're talking about! If the computer says SYNTAX ERROR, it usually means you spelled a word wrong, or forgot a word, or used a word the computer doesn't understand. It can also mean you used wrong punctuation: check your commas, semicolons, and colons. It can also mean you typed a left parenthesis that doesn't have a corresponding right parenthesis, or vice versa. It can also mean your DATA statement contains a string even though your READ statement says to read a number instead: to fix that problem, change the READ statement, by putting a dollar sign at the end of the variable's name.

In a statement such as PRINT 5+2, the + is called the *operation*; the 5 and the 2 are called the *operands*. If you try to say PRINT 5+2 but forget to type the 2, the computer will say MISSING OPERAND (or MO ERROR).

The biggest number the computer can handle is about 1E38 (which means 1 followed by 38 zeros). If you try to go much higher, the computer will say OVERFLOW (or OV ERROR). For example, if you say 1E39, or you try to multiply together lots of big numbers, you'll get an OVERFLOW ERROR.

If you try to divide by zero, the computer will say DIVISION BY ZERO (or /0 ERROR).

If the printer runs out of paper, the computer will say OUT OF PAPER.

When the computer sends a message to a device (such as a printer), the computer waits for the device to respond to the message. If a long time goes by and the computer still hasn't received any response from the device, the computer will say DEVICE TIMEOUT. For example, if you you try to use the printer and the computer says DEVICE TIMEOUT, it means the computer isn't receiving the proper feedback from the printer: that means the printer isn't properly attached to the computer, or hasn't been turned on, or isn't in a state of being READY.

If you feed the computer a number that's inappropriate, the computer will say ILLEGAL FUNCTION CALL (or FC ERROR). For example, if you say DELETE 40, but your program doesn't contain any line whose number is 40, the computer will say ILLEGAL FUNCTION CALL. If you feed the computer a negative number when the computer's expecting a positive number instead, the computer will say ILLEGAL FUNCTION CALL. For example, if you give the computer a negative number to find the square root of, or ask the computer to raise a negative number to a decimal power, or give the computer a subscript that's a negative number, the computer will say ILLEGAL FUNCTION CALL.

If you say GO TO 40, but your program doesn't contain any line whose number is 40, the computer will say UNDEFINED LINE NUMBER (or UL ERROR). The computer says UNDEFINED LINE NUMBER if you try to GO TO or GOSUB a missing line.

The computer handles two *types* of information: numbers and strings. If you feed the computer the wrong type of information—if you feed it a number when you should have fed it a string, or you feed it a string when you should have fed it a number—the computer will say TYPE MISMATCH (or TM ERROR). When you feed the computer a string, you must put the string in quotation marks; and you must put a dollar sign after the string's variable. If you forget to type a string's quotation marks or dollar sign, the computer won't realize it's a string; the computer will think you're trying to type a number instead; if the computer thinks a number would be inappropriate, the computer will say TYPE MISMATCH. So when the computer says

TYPE MISMATCH, it means you forgot a quotation mark or a dollar sign.

You can make the computer PRINT in two ways. One way is to type a line that begins with the word PRINT: the computer will obey such a line immediately (directly). The other way is to put a number before the word PRINT: that makes the PRINT statement a part of your program; the computer won't obey the statement until you type the word RUN. The word INPUT differs from the word PRINT: the word INPUT *must* have a number in front of it; it *must* be part of your program. If you forget to put a number in front of the word INPUT, the computer will say ILLEGAL DIRECT (or ID ERROR).

When you try to use the disks, you can make lots of errors. To use the disks, you must invent a name for your program; if the name you invent is inappropriate, the computer will say BAD FILE NAME (or FN ERROR). If you forget to put a disk into the disk drive, or you accidentally leave the disk drive's door open, the computer will say DISK NOT READY. If the disk's surface is scratched or otherwise damaged, the computer will say DISK MEDIA ERROR. If the disk is write-protected by a write-protect tab (which stops you from writing onto the disk) and you nevertheless try to write information onto the disk, the computer will say DISK WRITE PROTECT (or WP ERROR). If you try to write information onto the disk but the disk doesn't have enough free space to hold the information, the computer will say DISK FULL (or DF ERROR). The disk's *directory* is a place on the disk that remembers the names of all the disk's programs and files; if the directory is full, so that there's no more room to store names of programs, and you nevertheless try to put another program onto the disk, the computer will say TOO MANY FILES. If you tell the computer to copy JOE from the disk to the RAM (by saying LOAD "JOE" or RUN "JOE"), but the computer can't find JOE on the disk, the computer will say FILE NOT FOUND (or NON-EXISTENT or NE ERROR).

If you say READ, but the computer can't find any more DATA to read (because the computer has read all the DATA already), the computer will say OUT OF DATA (or OD ERROR).

Whenever you say FOR, you're supposed to say NEXT, and vice versa. If you say FOR without saying NEXT, the computer will say FOR WITHOUT NEXT. If you say NEXT without saying FOR, the computer will say NEXT WITHOUT FOR (or NF ERROR). If your program contains the words FOR and NEXT several times, but you accidentally have more FOR's than NEXT's or more NEXT's than FOR's, the computer will print one of those error messages, because the number of FOR's ought to equal the number of NEXT's.

When you use subscripts, you can make several kinds of errors. If the subscript is larger than the number indicated in the DIM statement, the computer will say SUBSCRIPT OUT OF RANGE (or BAD SUBSCRIPT or BS ERROR). If you give a DIM statement that contradicts another DIM statement (or you accidentally tell the computer to GO TO the same DIM statement twice), the computer will say DUPLICATE DEFINITION (or DD ERROR). You can also get those error messages if you forget to say DIM.

If your computer doesn't have enough RAM to hold your information, it will say OUT OF MEMORY (or OM ERROR). Part of the RAM is used for handling strings; if that part isn't big enough to hold all your strings, the computer will say OUT OF STRING SPACE (or OS ERROR). If you get one of those error messages, shorten your program (by splitting it into several smaller programs); or lower the number in your DIM statement; or (if you're using Radio Shack's model 3 BASIC) give a CLEAR command.

To make the computer do a subroutine, you're supposed to say GOSUB; and at the bottom of the subroutine, you're supposed to say RETURN. If you say RETURN but forgot to say GOSUB, the computer will say RETURN WITHOUT GOSUB (or RG ERROR). For example, if you accidentally say GO TO instead of GOSUB, the computer will say RETURN WITHOUT GOSUB. If you forget to say END at the bottom of your main routine, the computer will proceed from the main routine to the subroutine by a means other than GOSUB and will say RETURN WITHOUT GOSUB; in fact, the usual reason why a computer says RETURN WITHOUT GOSUB is that you forgot to say END at the bottom of your main routine.

PROGRAM JARGON

ELEMENTS

array: a variable that represents a list (or table) and has subscripts (176)
assignment statement: a statement that's an equation (58)

comment: a memo you insert in your program and that the computer ignores (115)
compound condition: a condition that contains the word AND or OR (72)
condition: what comes between the words IF and THEN (69)
consequence: what comes after the word THEN (69)
constant: either a numeric constant or a string constant (134)

dialect: a version of a computer language (34)
doubly subscripted array: a variable that represents a table (181)

E notation: a notation that means you should move the decimal point (38)
expression: a numeric expression or a string expression (134)

GW BASIC: gee-whiz BASIC; the fanciest version of Microsoft BASIC (34)

main routine: the main part of a program (184)
matrix: array (176)
Microsoft BASIC: the most popular dialect of BASIC (34)

nested subroutine: a subroutine whose RETURN goes to another subroutine (190)
numeric constant: a simple number, such as 8 or -524.6 or 1.3E5 (134)
numeric expression: a numeric constant or numeric variable or combination (134)
numeric variable: a letter (or notation) that stands for a number (58)

operation: addition, subtraction, multiplication, division, or exponentiation (191)

prompt: the question the computer asks, because of an INPUT statement (63)

random number: a number that's unpredictable (136)
remark: a memo you insert in your program and that the computer ignores (115)

string: a collection of characters, such as "I LOVE YOU" and "AIEEE LUF YA" (40)
string array: a variable that represents a list (or table) of strings (183)
string constant: a simple string in quotation marks, such as "I LOVE YOU" (134)
string expression: a string constant or string variable or combination (134)
string variable: a notation (such as X$) that stands for a string (61)
subroutine: an appendix at the bottom of a program (184)
subscript: a number such as the 2 in X_2, or the 3 in X(3) (176)

value: a number (or string) that something else stands for (58)
variable: either a numeric variable or a string variable (58)

zone: a part of the screen (128)

LOOPS & ENDS

abnormal exit: a line reached via a GO TO that skips from a loop (126)
abort your program: stop the computer from running your program (53)
counter: a variable that helps the computer to count (142)
decrement: decrease (141)
end mark: the words DATA END, which mark the end of the data (117)
end routine: a routine that the computer does, to end your program (117)
increment: increase (141)
infinite loop: a loop that the computer will repeat forever, unless aborted (53)
inner loop: a loop that's inside another loop (126)
limit value: the number that comes after the word TO, in a FOR statement (127)
loop: a part of your program that the computer does repeatedly (53)
nested loop: a loop that's inside another loop (126)
normal exit: the statement underneath the word NEXT (126)
outer loop: a loop that has another loop inside it (126)
READ loop: a loop whose top line says READ (116)
search loop: a loop whose purpose is to search for matching data (120)
step size: the number that comes after the word STEP, in a FOR statement (127)

GRAPHICS

gray level: a number that represents how dark a picture is at a point (212)
high resolution: dividing the screen into a very large number of pixels (140)
low resolution: dividing the screen into only a few pixels (140)
resolution: how many pixels your screen consists of (140)
stitching a curve: creating a curve, by drawing the curve's envelope (223)
X-coordinate: the first number in a pixel's name (140)
Y-coordinate: the second number in a pixel's name (140)

ERRORS

bug: an error in a program (112)
debug: remove bugs from your program (112)
error-handling routine: a routine that says to gripe about errors (69)
error trap: an error-handling routine (69)
garbage: incorrect input, or any other nonsense that gets into the computer (207)
gold-star program: a program that works perfectly on the first run (112)
overflow: a number too large for the computer to handle; larger than 1E38 (38)
syntax error: a command that the computer doesn't understand (47)
underflow: a number too tiny for the computer to handle; tinier than 1E-38 (38)

ADVANCED PROGRAMMING TECHNIQUES

algorithmic program: a program that finds a solution in all cases (281)
document your program: write an explanation of your program (208)
documentation: an explanation that helps people understand your program (208)
efficient program: a program that uses hardly any computer time or memory (200)
heuristic program: a program that finds a solution in most cases but not all (281)
means-ends analysis: the problem-solving approach used by GPS (282)
minimax method: a way to win chess, by drawing trees (256)
module: in a program, a bunch of consecutive lines that can't be punctured (198)
protocol method: writing a program by using a transcript of human thought (280)
pseudocode: a program line written in English instead of in computerese (198)
specification: making your goal precise (197)
top-down programming: writing a program by expanding a single line (198)

MEMORY JARGON

DISKS

Apple DOS: the DOS used on an Apple computer (29)
arm: the part of the disk drive that moves back and forth (84)

CP/M: a popular DOS that runs on every type 3 microcomputer (29)
CP/M-86: a DOS that works on the IBM Personal Computer (104)

disk: a medium that's faster, more convenient, and more reliable than tape (24)
disk cartridge: a round plastic cartridge that contains a hard disk (87)
disk command: a command that tells the computer to use the disk (29)
disk drive: what you put the disk into; it resembles a record player (24)
disk operating system: the collection of disk-command definitions (29)
disk pack: a stack of hard disks, and covered by a clear plastic cylinder (87)
door: the part of the disk drive that covers the slit (24)
DOS: disk operating system (29)
double-density disk: a disk that's guaranteed to handle double-density DOS (85)
double-density DOS: a DOS that puts twice as much info on a disk's side (85)
double-sided disk: a disk that's guaranteed to work on both sides (85)
double-sided DOS: a DOS that uses both sides of the disk (85)
DSDD: double-sided double-density (85)

fixed-head disk drive: a disk drive in which each track has its own head (87)
floppy disk: a disk made of thin material that bends easily (24)
flying-head disk drive: a disk drive in which the head flies above the disk (87)
format the disk: divide the disk's surface into tracks and sectors (85)

hard disk: a disk made of thick material that's hard to bend (24)
hard-sectored floppy disk: a floppy disk that has many tiny holes in it (85)

jacket: the black square envelope that permanently covers the floppy disk (24)

KILL: a command that makes the computer erase your program from the disk (111)

landing strip: in a flying-head drive, where the head lands (87)
LOAD: a command that copies your program from the disk to the RAM (111)

moving-head disk drive: a disk drive in which several tracks share a head (87)
MS-DOS: the DOS from which PC-DOS was developed (29)

non-removable hard disk: a hard disk that can't be removed from the drive (87)

pack: usually, a disk pack (87)
PC-DOS: the DOS used on IBM's Personal Computer (29)

removable hard disk: a hard disk that you can remove from the disk drive (87)
runs under CP/M: requires that CP/M be in the RAM beforehand (29)

SAVE: a command that copies your program from the RAM to the disk (110)
sector: a part of a disk's track (84)
single-density disk: a disk that might not handle a double-density DOS (85)
single-density DOS: a DOS that puts a normal amount of info on a disk's side (85)
single-sided disk: a disk that's been tested on only one side (85)
single-sided DOS: a DOS that uses just one side of the disk (85)
soft-sectored floppy disk: a floppy disk that has just one tiny hole in it (85)
spindle: the turntable inside the disk drive (24)
SSDD: single-sided double-density (85)
SSSD: single-sided single-density (85)
system master: a disk that contains the definitions of all the disk commands (29)

tab: usually, a write-protect tab (86)
track: a circle that the head draws on the disk's surface (84)
track 0: the track closest to the disk's outer rim (84)
TRSDOS: the DOS used on a Tandy Radio Shack computer (29)

utility: a helpful program that's on the system-master disk (29)

Valdocs: the DOS designed for Epson's QX-10 microcomputer (104)

Winchester disk drive: IBM's first kind of flying-head disk drive (87)
write-protect notch: a notch in a floppy disk (86)
write-protect tab: a plastic tab that can cover the write-protect notch (86)
write-protected disk: a disk that's protected from being written on (86)

TAPES

audio cassette: an ordinary cassette you put into an ordinary tape recorder (82)
digital tape cartridge: a cartridge containing tape used just for computers (82)
reel-to-reel tape drive: a tape drive that uses a huge reel of tape (82)
Stringy Floppy: the digital tape cartridge system manufactured by Exatron (82)
tape drive: a tape recorder designed especially for use by computers (82)

MEMORY CHIPS

firmware: software that's permanently etched into the computer's ROM (31)
memory board: an extra PC board that contains memory chips (79)
memory card: a PC card that's a small memory board (79)
memory cartridge: a cartridge that contains a memory card (79)
memory chip: a chip whose purpose is to remember information (77)
RAM: random-access memory (23)
Ramdisk: a fake disk that consists of RAM (80)
random-access memory: in the circuitry, the part that remembers the problem (23)
read-only memory: an electronic circuit that holds information permanently (28)
ROM: read-only memory (28)
Semidisk: a fake disk that consists of RAM (80)
Solid-State Disk: a fake disk that consists of RAM (80)

GENERAL TERMS

backup: a copy of a tape or disk, and used only if the original is damaged (83)
byte: a character; any symbol you can type on the keyboard (80)
cartridge: a memory cartridge or digital tape cartridge or disk cartridge (79)
drive: either a tape drive or a disk drive (82)
G: gigabyte (80)
gigabyte: about a billion bytes; more precisely, it's 1,073,741,824 bytes (80)
K: kilobyte (80)
kilobyte: about a thousand bytes; more precisely, it's 1024 bytes (80)
M: megabyte (80)
meg: megabyte (80)
megabyte: about a million bytes; more precisely, it's 1,048,576 bytes (80)
memory: the part of the computer that remembers (79)
read-write head: the "needle" that reads and writes on a tape or disk (84)
software: information inside the computer (31)

SHARE

Share your secret: photocopy the next page, and give it to a friend. If you know a young (or young-at-heart) computerist who's having a birthday, Christmas party, or graduation, turn these volumes into a present: send me the computerist's name, address, and either a greeting card or what you want me to say.

PRICES & POLICIES

The Secret Guide to Computers consists of two volumes. Volume 1 (The Main Secrets) covers everything important about computers; volume 2 (Deep Secrets) digs even deeper. Each volume costs just $14. Do *not* send money for regular (book-rate) postage and handling: I'll pay for them myself, even if you live in a foreign country. If you're not sure whether to order a book, go ahead: you can return unused books at any time, for a 100% refund.

DISCOUNTS

The list price is $14 per book; so the two-volume set costs $28. To pay less, get together with a friend:

20% discount if you order 2 sets (or any 4 books); you pay just $11.20 per book
30% discount if you order 4 sets (or any 8 books); you pay just $9.80 per book
40% discount if you order 10 sets (or any 20 books); you pay just $8.40 per book
50% discount if you order 50 sets (or any 100 books); you pay just $7.00 per book
60% discount if you order 500 sets (or 1000 books); you pay just $5.60 per book

START YOUR OWN BOOK BUSINESS

To start your own book business, buy books in quantity from us at discount, and sell them to your friends for slightly under list price. You'll make your friends happy (since they'll pay less than list price), and you'll be happy also (because you'll be helping your friends and also be turning a profit).

IF YOU'RE IN MASSACHUSETTS . . .

If the person who orders the book, or the person we're shipping the book to, is in Massachusetts, we must charge Massachusetts sales tax—unless the book is for a government agency or tax-exempt agency (give us you tax-exempt number) or for resale (give us your Massachusetts resale number) or is a *required text* for a course (tell us which school and course).

PICK-UP

Normally, you get the books by mail. If you live near Boston and don't want to wait for your mailman, come to us and pick up the books yourself: we're near Prudential Center. Before you come, phone us at 617-266-8128, so we can give you directions and a guaranteed pick-up time.

PURCHASE ORDERS

If you work for a government agency (such as a public library or a public school), or an accredited private school, or an established bookstore or computer store, or a major computer manufacturer, we'll accept your purchase order, which you must pay 30 days after receiving the books.

REVIEW COPIES

If you're a reviewer for a large magazine, or the head of a large computer department, you might be able to get a complimentary copy of The Secret Guide to Computers for review. To find out, phone us.

CUT OUT THIS COUPON!

How many copies do you want of volume 1 (The Main Secrets)? _____

How many copies do you want of volume 2 (Deep Secrets)? _____

TOTAL number of books: _____

TOTAL number of books	Price per book, usually	Price per book, if you're a Massachusetts resident (including 5% sales tax)
1 - 3	$14.00	$14.70
4 - 7	$11.20	$11.76
8 - 19	$9.80	$10.29
20 - 99	$8.40	$8.82
100 - 999	$7.00	$7.35
1000 & up	$5.60	$5.88

Multiply your TOTAL number of books by the price per book, and write the TOTAL book price here:

Put a check mark in front of the shipping method you choose. . . .

□ Book rate (or library rate or parcel post or surface), to anywhere in the world: we charge nothing; free!

□ Special handling (which is faster than book rate, and almost as fast as priority mail), to U.S. only: we charge just $1 total, even if your order is large.

□ Priority mail (first-class air-mail): we charge $1.50 *per book* to New England, $2 *per book* to other continental U.S. east of the Mississippi, $2.50 *per book* to other U.S., $6 *per book* to other countries.

FREE brochures on the following topics will become available in April 1984. You can get on our mailing list now, by checking the brochures you want. . . .

□ how you can get later editions of The Secret Guide to Computers

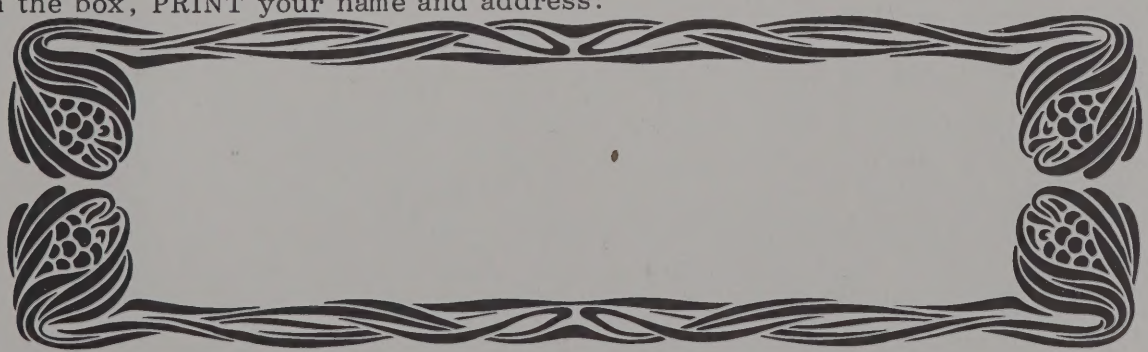
□ how you can get special editions of The Secret Guide to Computers for use by young children and for students in elementary, junior-high, and high schools

□ how you can get The Secret Guide to Tricky Living (part 1 is "Sex"; part 2 is "Everything But")

□ how you can attend one-day and two-day courses taught by Russ in various cities

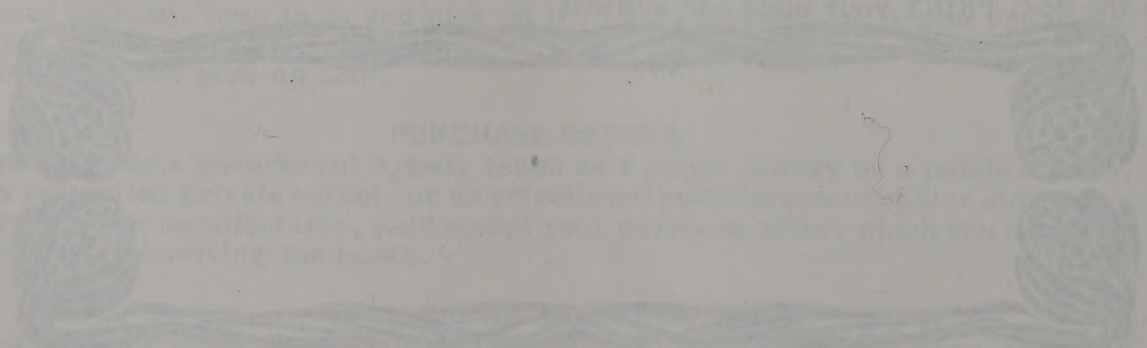
□ how you can attend one-week, four-week, and six-week courses taught by Russ's protégés, at various colleges in the summer

In the box, PRINT your name and address:



On the back, write any comments you have (pro or con) about The Secret Guide to Computers.

Add the total book price to the shipping charges (if any); make the check or money order payable to The Secret Guide to Computers. Send to Russ Walter, 92 Saint Botolph St., Boston, MA 02116.



TEN REASONS *why* THE SECRET GUIDE TO COMPUTERS *is such a popular set:*

SIMPLE TO READ. *Particularly popular among kids, foreign students, and adults who want straightforward talk.*

FAST-PACED. *By reading 10 pages of The Secret Guide, you'll learn as much as by reading 50 pages of most competing texts.*

FOOLPROOF. *Each edition has been carefully tested on beginners, and revised to make sure you can't go wrong.*

COMPLETE. *This two-volume set teaches you every aspect of computers and the computer industry.*

INSIDE INFO. *Anecdotes from programmers' locker rooms. Tricks the experts use, to make programming easy.*

EXAMPLES GALORE. *They're easy to type, and ready for you to run.*

BACK-UP HELP. *Call the author, at 617-266-8128, whenever you have a question about computers.*

RIDICULOUSLY LOW PRICE. *Costs less than any other comprehensive computer text.*

NO MATH BACKGROUND REQUIRED, BEYOND FOURTH GRADE. *The volumes explain all the tools for advanced math.*

IT'S FUN!